Apple PRESS

San Francisco Apple Core

IN THIS ISSUE

Notices 1
B.A.S.I.C. - Joyce Conklin 2
FORTH - Dudley Ackerman 6
Pascal
  Center Formatting - Max J. Nareff 10
  Avoiding Array Disarrays - Max J. Nareff 11
  Interrupts on the Apple II - Max J. Nareff 12
  Easy I/O Routines - Michael Fischer 15
F.M.L.I. - Bryan Costales 18
VisiCalc & Your Fed Tax - Don Scellato 28
VisiCalc Interest Groups - Don Scellato 32
Nu'ers - Kathy Rudovsky 34
Marketing - Don Scellato 35
Membership Application 36

GENERAL MEETINGS:

Macintosh Demonstration! - March 17
April 14

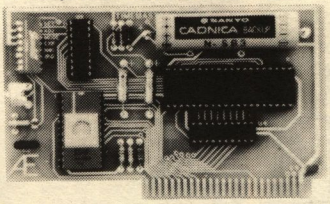
FEBRUARY/MARCH, 1984

APPLIED ENGINEERING Quality Starts On The Drawing Board....

But it doesn't end there. Our commitment to excellence applies all the way down the line, through manufacturing and testing to our three year no-hassle warranty. The aim, of course, is to provide you with the best designed and best crafted hardware available. Try any of the peripherals below and see what it's like to own the finest.

The TIMEMASTER
Finally, a clock that does it ALL!

Automatically date stamps files with PRO-DOS



Fully PRO-DOS compatible

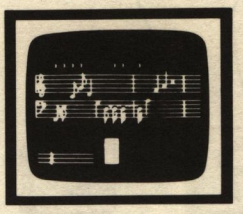
- Just plug it in and your programs can read the year, month, date, day, and time to 1 millisecond! The only clock with both year and ms.
- A rechargeable NiCad battery will keep the TIMEMASTER running for over ten years.
- Powerful 2K ROM driver — No clock could be easier to use.
- Full emulation of most other clocks, including Mountain Hardware's Appleclock (but you'll like the TIMEMASTER mode better).
- Basic, Machine Code, CP/M and Pascal software on 2 disks!
- Eight software controlled interrupts so you can execute two programs at the same time (many examples are included).
- On-board timer lets you time any interval up to 48 days long down to the nearest millisecond.

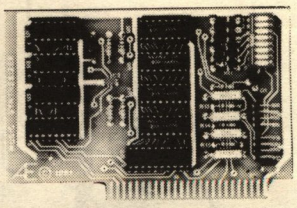
The TIMEMASTER includes 2 disks with some really fantastic time oriented programs (over 25) plus DOS dater so it will automatically add the date when disk files are created or modified. The disk is over a $200.00 value alone — we give the software others sell. All software packages for business, data base management and communications are made to read the TIMEMASTER.

If you want the most powerful and the easiest to use clock for your Apple, you want a TIMEMASTER.

PRICE $129.00

Super Music Synthesizer
Improved Hardware and Software

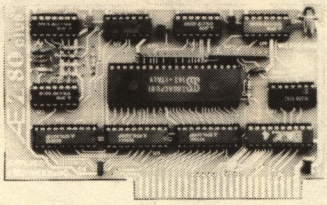




- Complete 16 voice music synthesizer on one card. Just plug it into your Apple, connect the audio cable (supplied) to your stereo, boot the disk supplied and you are ready to input and play songs.
- It's easy to program music with our compose software. You will start right away at inputting your favorite songs. The Hi-Res screen shows what you have entered in standard sheet music format.
- Now with new improved software for the easiest and the fastest music input system available anywhere.
- We give you lots of software. In addition to Compose and Play programs, 2 disks are filled with over 30 songs ready to play.
- Easy to program in Basic to generate complex sound effects. Now your games can have explosions, phaser zaps, train whistles, death cries. You name it, this card can do it.
- Four white noise generators which are great for sound effects.
- Plays music in true stereo as well as true discrete quadraphonic.
- Full control of attack, volume, decay, sustain and release.
- Will play songs written for ALF synthesizer (ALF software will not take advantage of all our card's features. Their software sounds the same in our synthesizer.)
- Our card will play notes from 30HZ to beyond human hearing.
- Automatic shutoff on power-up or if reset is pushed.
- Many many more features.

PRICE $159.00

Z-80 PLUS!



- TOTALLY compatible with ALL CP/M software.
- The only Z-80 card with a special 2K "CP/M detector" chip.
- Fully compatible with microsoft disks (no pre-boot required).
- Specifically designed for high speed operation in the Apple IIe (runs just as fast in the II+ and Franklin).
- Red "CP/M WORKING" LED indicator, the Z-80 Plus does not interfere with non-CP/M programs.
- A semi-custom I.C. and a low parts count allows the Z-80 Plus to fly thru CP/M programs at a very low power level. (We use the Z-80A at fast 4MHZ.)
- Does EVERYTHING the other Z-80 boards do, plus Z-80 interrupts.

Don't confuse the Z-80 Plus with crude copies of the microsoft card. The Z-80 Plus employs a much more sophisticated and reliable design. With the Z-80 Plus you can access the largest body of software in existence. Two computers in one and the advantages of both, all at an unbelievably low price.

PRICE $139.00

Viewmaster 80

There used to be about a dozen 80 column cards for the Apple, now there's only ONE.

- TOTALLY Videx Compatible.
- 80 characters by 24 lines, with a sharp 7x9 dot matrix.
- On-board 40/80 soft video switch with manual 40 column override
- Fully compatible with ALL Apple languages and software—there are NO exceptions.
- Low power consumption through the use of CMOS devices.
- All connections on the card are made with standard video connectors.
- All new design (using a new Microprocessor based C.R.T. controller) for a beautiful razor sharp display.
- The VIEWMASTER incorporates all the features of all other 80 column cards, plus many new improvements.

	PRICE	BUILT IN SOFTSWITCH	SHIFT KEY SUPPORT	LOW POWER DESIGN	THREE YEAR WARRANTY	7x9 DOT MATRIX	LIGHT PEN INPUTS	40 COLUMN OVERRIDE	INVERSE CHARACTERS
VIEWMASTER	**169**	**YES**	**YES**	**YES**	**YES**	**YES**	**YES**	**YES**	**YES**
SUP'RTERM	MORE	NO	YES	NO	NO	NO	NO	YES	YES
WIZARD80	MORE	NO	NO	NO	NO	YES	NO	YES	YES
VISION80	MORE	YES	YES	NO	NO	YES	NO	NO	NO
OMNIVISION	MORE	NO	YES	NO	NO	NO	NO	YES	YES
VIEWMAX80	MORE	YES	YES	NO	NO	YES	NO	NO	YES
SMARTERM	MORE	YES	YES	NO	NO	NO	YES	YES	NO
VIDEOTERM	MORE	NO	NO	YES	NO	YES	YES	NO	YES

The VIEWMASTER 80 works with all 80 column applications including CP/M, Pascal, WordStar, Format II, Easywriter, Apple Writer II, VisiCalc, and all others. The VIEWMASTER 80 is THE MOST compatible 80 column card you can buy at ANY price!

PRICE $169.00

MemoryMaster IIe 128K RAM Card

- Expands your Apple IIe to 192K memory.
- Provides an 80 column text display.
- Compatible with all Apple IIe 80 column and extended 80 column card software (same physical size as Apple's 64K card).
- Can be used as a solid state disk drive to make your programs run up to 20 times FASTER (the 64K configuration will act as half a drive).
- Permits your IIe to use the new double high resolution graphics.
- Automatically expands Visicalc to 95 K storage in 80 columns! The 64K config. is all that's needed, 128K can take you even higher.
- PRO-DOS will use the MemoryMaster IIe as a high speed disk drive.
- Lowest power consuming 128K card AVAILABLE.
- Complete documentation included, we show you how to use all 128K.

If you already have Apple's 64K card, just order the MEMORYMASTER IIe with 64K and use the 64K from your old board to give you a full 128K. (The board is fully socketed so you simply plug in more chips.)

MemoryMaster IIe with 128K	**$249**
Upgradeable MemoryMaster IIe with 64K	**$169**
Non-Upgradeable MemoryMaster IIe with 64K	**$149**

Our boards are far superior to most of the consumer electronics made today. All I.C.'s are in high quality sockets with mil-spec. components used throughout. P.C. boards are glass-epoxy with gold contacts. Made in America to be the best in the world. All products work in the APPLE IIE, II, II+ and Franklin. The MemoryMaster IIe is IIe only. Applied Engineering also manufactures a full line of data acquisition and control products for the Apple; A/D converters and digital I/O cards, etc. Please call for more information. All our products are fully tested with complete documentation and available for immediate delivery. All products are guaranteed with a no hassle **THREE YEAR WARRANTY.**

All Orders Shipped Same Day
Texas Residents Add 5% Sales Tax
Add $10.00 If Outside U.S.A.
Dealer Inquiries Welcome

Send Check or Money Order to:
APPLIED ENGINEERING
P.O. Box 470301
Dallas, TX 75247

Call (214) 492-2027
7a.m. to 11p.m. 7 days a week
MasterCard, Visa & C.O.D. Welcome
No extra charge for credit cards

CIDER PRESS

MULTIPLAN

New Multiplan user with Apple //e needs to talk to someone who is proficient with the program. Please call Sarah Collings at 415-383-9631 (Eve) or at 408-249-7550 (Days).

COMPUTER SALES

Due to recent expansion we have exciting new opportunities for proven sales professionals in retail and direct sales. You will receive strong technical and sales training on our complete product lines that include Apple's new Macintosh, Texas Instrument Professionals, Epson QX-10, IBM compatibles, and many others.

Preference will be given to people with computer sales experience. Compensation includes guarantee plus commission with bonus.

Join our winning team. Full and Part Time employees will be considered. Send resume and salary history to:

Christopher Aitken
Sales Manager
The Computer Store
1320 Webster Street
Oakland, Ca.
94612

MODEMS

Looking for someone good with modems to help me transfer files from one group to another. Will provide details to interested parties. Call Don Chan 982-0616 (Weekday days)

INCREDIBLE JACK

User of INCREDIBLE JACK word processing program needs to talk to experienced user. Please call Dan Loomis at 415-435-3270

THE CORE'S CORE

DIRECTORS:

Sami Capron	415/388-8631
Joyce Conklin	415/341-6955
Michael Fischer	415/933-4435
Don Scellato	415/672-2051
Alan L Smerican	415/566-2342
Allan B. Woods	415/687-9659
Alan Morris	

MEMBER SERVICE/COMMITTEES:

Membership
Alan L Smerican 415/566-2342
Ted Chatterton

Cider Press -		
Editor	Allan Woods	415/687-9659
BASIC	Joyce Conklin	415/341-6955
Marketing Dir.	Don Scellato	415/672-2052
Review Group	Ben Lai	415/564-7773
General Meeting	Michael Fischer	415/933-4435
Coordinating Director	Joyce Conklin	415/341-6955
Treasurer	Alan Morris	

Computer Graphics - - - - Mei-Ying Chen

CIDER PRESS POLICY

All manuscripts, photographs, and other materials will be considered submitted free and released for publication. They become the property of the San Francisco Apple Core and the CIDER PRESS. Authors should clearly mark all material submitted for publication so that proper credit can be given.

The views expressed in any articles are those of the authors, and not necessarily those of the San Francisco Apple Core, Inc., the Editor, Club Officers or Directors . . . or anyone else in their right minds.

The San Francisco Apple Core Incorporated is a non - profit organization comprised of and supported by owners and users of Apple I I and Apple / / / computers. The Club is run by officers and committees, and the Club endeavors to aid Apple owners through its Program Libraries, Program Disks, meetings, and this publication.

Article Submission: The CIDER PRESS welcomes any and all articles dealing with Apple computers and their associated hardware and software. Articles may be submitted (typed or printed, double - space), or as disk text files using any of the popular word processors or text editors. Any author who has an article published in the CIDER PRESS will receive a free copy of the most current Disk - of - the - Month.

Reprints: All articles appearing in the CIDER PRESS not copyrighted by the author may be reprinted by other *non - profit* Apple User Groups, without prior permission, so long as proper credit is given to both the author and the CIDER PRESS. Permission to reprint a copyrighted article must be obtained in advance, by writing to the authors, c/o the San Francisco Apple Core.

San Francisco Apple Core
1515 Sloat Blvd.
San Francisco, CA 94132

B*A*S*I*C

Copyright (c) 1983 by Joyce Conklin

Kenny Rogers, Watch Out!

'Way back last spring we started working on a poker game as a vehicle for a discussion of arrays. The two-dimensional array was established with 13x4 elements to give a place for each card (no jokers). By making the array 14x5 instead, you can keep track of how many cards of one suit (the 14th element of the first dimension) or how many cards of one rank (the 5th element of the second dimension) have been "dealt" to your hand. The reason I haven't continued with this discussion is that I have been trying to come up with a more elegant algorithm than a series of IF...THEN lines for checking the value of the hand. No luck so far. The algorithm I used in my first programming class went like this (I don't have the exact listing to hand, so I will try to reconstruct it from notes I have been making whenever I have had a little time to work on it; the values in the 14th column and the 5th row need to be accumulated as the cards are dealt):

```
1000 FOR J = 1 TO 4

1010 IF D(14,J) = 5 THEN FLUSH = 1 AND
     SUIT = J
     REM value of 1 indicates 5 cards of
         all the same suit and J records
         which suit

1020 NEXT J

1030 FOR K = 13 TO 10 STEP - 1

1040 IF D(K,5) = 1 THEN ROYAL =
     ROYAL + 1

1050 NEXT K

1060 IF FLUSH = 1 AND ROYAL = 5 THEN
     ROYFL = 1 : GOTO <print-out sub-
     tine>
```

Notice in the above line I have included something that definitely is NOT included in your standard Applesoft BASIC! This is my way of indicating that you will have to plan the line number you want to use for the area of your program which will tell the user the value of the hand dealt, but I don't have the line numbers planned for my own program. One more reason to have your program written out on paper with either a flow-chart or pseudo-code before entering it into the computer's memory.

Before continuing, I'd better illustrate the conception of the array that I am thinking about so you can see what is being checked in the following lines. The array is dimensioned 14 x 5 so that 13 "boxes" (elements, in computereze) are set aside for the cards in a suit. I conceive of element #1 being the deuce, element #2 the trey, #3 the four, etc. If you get confused by this, just make your array bigger and don't use the first (0) and second (1) elements. That way element #2 is the deuce, #3 the trey, etc. Mentally I think of the array being 14 elements horizontally by 5 elements vertically. The easiest way I have found to explain arrays to students is to have them think about the faculty mailboxes, which are stacked up in the main office. If you are not school-oriented, think about the mailboxes in the Post Office. The 14th "column" of boxes will have 5 stacked one on top of the other; we will think about the top four representing the four suits. I think of it as Spades, Hearts, Diamonds and Clubs (top to bottom) but you can use any order you want.

```
1070 FOR J = 1 TO 8

1080 FOR K = J TO J+5

1090 IF D(J) = 1 THEN ST = ST + 1

1100 NEXT K

1110 IF ST = 5 THEN STRAT = 1 : J = 9 :
     GOTO <print-out routine>
```

If you have 5 cards in a row, that's a straight. You record that fact (STRAT = 1), set J to the value which would finish the loop and only then go out of the loop. Maybe this isn't strictly necessary, but I have run into trouble when dropping out of a FOR...NEXT loop before it's finished. There's even more trouble if you are prone to "spaghetti code" (lots of GOTO's hopping back and forth all over the place) and you try to re-enter a loop that was left unfinished.

```
1120 ST = 0 : NEXT J
     REM If you didn't have a straight
         6-high, you have to reset the
         straight-counter (S=0) before
         checking for a 7-high straight,
         etc.
```

```
1130 FOR J = 1 TO 13

1140 IF D(J,5) = 4 THEN FOUR = 1 :
     J = 13 : GOTO <print-out routine>

1150 IF D(J,5) = 2 THEN PAIR = PAIR + 1

1160 IF D(J,5) = 3 THEN TRIPLE = 1

1170 NEXT J

1180 IF PAIR = 1 AND TRIPLE = 1 THEN
     FULL = 1 : GOTO <print-out routine>

1190 IF PAIR = 0 THEN ZIP = 1 : GOTO
     <print-out routine>
```

Your print-out routine will then check the values of the various counters/flags in order to determine the winning hand (if you've added that many bells and whistles to your program). The order for the check would be:

ROYFL = 1? (Royal flush - you would have this condition if you left the tally module after line 1060.

FLUSH = 1 AND STRAT = 1? (Straight flush)

FOUR = 1? (Four of a kind)

TRIPLE = 1 AND PAIR = 1? (Full house)

FLUSH = 1? (Flush)

STRAT = 1? (Straight)

TRIPLE = 1? (Three of a kind)

PAIR = 2? (Two pair [No kidding!!])

PAIR = 1? (One pair)

ZIP = 1? (Ooops - sorry!)

Each of these checks would require an IF...THEN statement and a PRINT following when the check is true. Maybe now you see what I meant about the lack of elegance. Any better programs out there waiting for publication? Send 'em to the Cider Press!

. . . . .

A Guest Lecturer

Morris Prosser is back again, with answers to the questions in the October column. Since he does such a complete job of explaining, I will reproduce his answers here verbatim, rather than trying to paraphrase them.

1. Regarding my question concerning the formatting of the listing of a program:

"As you know, Applesoft commands are replaced by tokens, which have the high bit set, [J.C.: that is different from the ordinary ASCII code which only uses 7 bits and leaves the 8th or high bit at zero] for storage in memory. Literal characters, such as variable names or characters in REM, PRINT, DATA, or INPUT statements, are stored with the high bit cleared [set to zero]. Spaces are not stored unless they are part of REM, PRINT, DATA, or INPUT statements.

"Of course, not storing spaces saves a lot of memory, but a listing without spaces is pretty hard to read [J.C.: ever try to read one from a TRS-80?], so spaces are inserted at certain points in the listing process. It happens that literal characters are printed verbatim, with no spaces added, but tokens are converted into keywords and printed with spaces both before and after, so they will always be separated from the literal characters and from each other. For example, if you enter a line

"10PRINTA"

and list it, it will be listed as

"10 PRINT A"

- notice the one space between "PRINT" and "A". If you enter a line

"20PRINTPEEK(A)"

it will list as

"20 PRINT PEEK (A)"

- two spaces between "PRINT" and "PEEK", one following "PEEK".

"In listing a program, Applesoft always prints a space after the line number. Whether an additional space is printed deends only on whether the first item in the line is a variable (a literal character) or a keyword. I can't at the moment think of anything other than a keyword or a variable than can be the first thing in an Applesoft line without causing a syntax error."

2. "In the September column [J.C.: Cider Press, Oct. 1983,page 8] you asked a question about the algorithm you were using to group scores into six categories. The situation is even worse than you thought! [Ain't that always the case?!! J.C.] As printed, the algorithm does not include perfect scores (Q = 7), and any score less than 30% results in a negative Q - not a zero - so it would be lost as well. Entering an ON...GOTO statement with a value of zero does not result in an error message - program execution simply falls through to the next statement. The error message occurs when Q is negative. I was not interested in the particular application, [J.C.: that's the problem - the public isn't interested enough in education!! - Just kidding, Morris!] but the algorithm for all cases was an interesting exercise. I suggest the following:

```
520 Q = INT ((SC(L) / N) * 10 + .05) - 3:
    Q = Q * (Q < 0) * (Q < 7) + (Q < 1) +
    6 * (Q < 6) : FD(Q) = FD(Q) + 1
```

"Now look at the next few lines (530-580). By using the ON...GOTO, all the lines after the destination line are executed, so if Q = 1, FD(1) through FD(6) are all incremented! This is not a good way to go, unless your object is to demonstrate the ON-GOTO command, and even then it needs work. The algorithm I suggested replaces lines 515 through 580."

3. Finally, for those who are interested in learning more about the innards of the Apple (is that the Apple Core?), an answer about the ESC function in PLE (Program Line Editor):

"You asked another question, too - about the way PLE partially disconnects after using "PR#0", then reconnects after using CTRL-E to edit a line. There is a location in DOS ($AA53.AA56) that holds the addresses of the true input and output handlers. If you go to the monitor and look at these, you will find $9B3E for the output handler and $9723 for the input handler. Look at these addresses after using "PR#0" and you will see that the output handler address is $FDF0, which sends output to the screen rather than to PLE. Input is still coming through PLE, but if you now type "IN#0", the input handler address will change to $FD1B, which gets input from the keybord. Typing CTRL-Y from the monitor or "&" form Applesoft calls a routine in PLE that reconnects its input and output addresses.

"If you have used "PR#0" and not "IN#0", input is still coming through PLE and when you use CTRL-E to edit a line, it goes through a routine that resotres $9B3E as the output handler address. The part of the code that accepts ESC as a pause signal like CTRL-S is in the output handler - it is the PLE version of VIDWAIT, and accepts either ESC or CTRL-S for a pause in the listing. That of course is the answer to the specific question you asked."
[J.C.: Don't you just love the "of course" in the above? Reminds me of a tutor I had once in my one and only calculus course - he was always saying "Obviously ...." when it wasn't obvious to me at all!!]

All kidding aside, Morris - many thanks for the answers to the questions. Maybe you should think about doing a guest shot next summer to let me go travelling for a while.

. . . . .

Got an Answer?

Now for this month's puzzler. I needed to generate several random order lists of numbers for use in my classes. The students become very adept at knowing when their "turn" is coming if I stick to alphabetical order and will knuckle down and study only when their "number is up" (or when a test is imminent!). I wanted to have two-three lists which I could use concurrently so they needed to be displayed in parallel columns which would be placed under the acetate cover of my blotter pad. Here's the program to generate three lists with different order, but all having numbers in the same range:

```
10 HOME

20 X = RND ( PEEK (78) + PEEK (79) *
    256 ) : X = RND ( -X )

30 INPUT "NUMBER = "; N :
   REM This will be the top number for
       range; the number of students in
       the class

40 DIM A(N) : DIM B(N) : DIM C(N)

50 FOR L = 1 TO N : A (L) = L : B(L) =
   L : C(L) = L : NEXT

60 FOR J = 1 TO N/2 :
   REM This will make N/2 "shuffles of
       the deck" - if this doesn't mix
       them well enough, just do it all
       the way to N

70 R1 = INT( RND (1) * N) + 1 : R2 = INT
   ( RND (1) * N) + 1 : R3 = INT ( RND (
   1) * N) + 1 : R4 = INT ( RND (1) * N)
   + 1 : R5 = INT ( RND (1) * N) + 1 :
   R6 = INT ( RND (1) * N) + 1

80 T = A(R1) : A(R1) = A(R2) : A(R2) = T:
   T1 = B(R3) : B(R3) = B(R4): B(R4) =
   T1: T2 = C(R5) : C(R5) = C(R6): C(R6)
   = T2

90 NEXT J

100 FOR K = 1 TO N : PRINT A(K),B(K),
    C(K) : NEXT K
```

Now, at this point, I would have a neat set of scrambled numbers lined up, but notice that they would still be displayed on the screen, which would really leave a lump on my desk to say nothing of an empty spot on my computer table at home! Obviously I must insert a command to turn on the printer. I figured that since there was only one PRINT statement (line 100) that anywhere above that was O.K. so I put it right up at the top (line 5) -- that didn't work at all! But if I placed it at line 95 (e.g. 95 PR#1) and then 110 PR#0, everything was hunky-dory. Is it the presence of HOME in line 10 that caused the problem?

. . . . .

Renumber Revisited

A call from Helen Phillips, a new member of SFAC who owns a //e, alerted me to a "problem" which isn't one for the old-timers with the II+ System Master diskette. The RENUMBER program on the //e System Master has had its (in)famous bug fixed so that it no longer causes randumb (sic) changes in multipliers which happen to fall within the range of the renumbering, but the old program called RENUMBER INSTRUCTIONS no longer exists, at least in its expanded format. So, as a service to those //e owners who want to learn more about the programming capabilities of their machine, herewith a somewhat edited version of those RENUMBER INSTRUCTIONS.

WARNING!! Do not press Ctrl-Reset during renumber or merge - it will destroy your program!! Also do not use 'MAXFILES'; it will destroy RENUMBER! [In other words, MAXFILES sets aside memory for file buffers that is normally used for the RENUMBER program].

This program can renumber all or part of a BASIC program. By typing "& <RETURN>" it will renumber the entire program starting at ten and going by tens.

```
& FIRST 50, INC 5
```

FIRST sets the first new line number, INC sets the step to the next line number. You can set the new first line and line increment to almost any value. [Don't use line #0 - it causes Applesoft to hang up.]

To renumber part of a program, you must specify which part by START and END; START sets the first line and END sets the last line of your current program which will be renumbered. For example, if your original program looked like this:

```
2 REM THIS IS A SAMPLE PROGRAM
4 PRINT "WHAT FUNNY LINE NUMBERS"
6 FOR L = 1 TO 1000
8 X = L^2
10 Y = L*L
12 NEXT L
14 PRINT "A SAMPLE OF INTERNAL CALCULA
   TION DIFFERENCES"
16 PRINT X, Y
18 PRINT : PRINT "THE ABOVE NUMBERS
   SHOULD BE THE SAME"
20 END
```

If you decide that the line numbers are too close together, but you don't really need to renumber the REM statement, you could do the rest of the program like this:

```
& F300, I10, S4, E20
```

(In this case, the E20 is optional; if not specified, RENUMBER will go all the way to the last program line.)

```
2 REM THIS IS A SAMPLE PROGRAM
300 PRINT "WHAT FUNNY LINE NUMBERS"
310 FOR L = 1 TO 1000
320 X = L^2
330 Y = L*L
340 NEXT L
350 PRINT "A SAMPLE OF INTERNAL CALCULA
    TION DIFFERENCES"
360 PRINT X, Y
370 PRINT : PRINT "THE ABOVE NUMBERS
    SHOULD BE THE SAME"
380 END
```

Notice that the program only recognizes the initial letter (F, I, S, or E) rather than the entire word. Extra letters and spaces after a command are ignored.

This program can also merge the BASIC program in the hold file to the BASIC program in memory. First you must LOAD the first program and type "&H". Next, load the second program. You may RUN or RENUMBER this program as needed. Then, to merge the two, type "&M". MERGE will combine the hold file with the program currently in memory. Line numbers are arranged in ascending order. **Duplicate line numbers will both appear, hold version second.** Merging with no program in memory restores the hold file.

Input syntax: The first letter after '&' must be E, F, H, I, M or S. [The documentation also mentions C for convert, which changes a RAM Applesoft program to run with a ROM card and vice versa, but relatively few people do that - I know I never have!] H and M execute immediately; the rest of the line is ignored. E, F, I, and S set up renumber parameters which are decimal numbers in the range of 0 to 63999. If no number follows E, F, I or S, then the parameter is set to 0. Multiple parameters are separated by commas and may be in any order.

Most error messages are self-explanatory and will stop execution before any changes are made to the BASIC program.

RENUMBER fixes line number references in these statements: GOTO, ON...GOTO, GOSUB, ON...GOSUB, DEL, LIST, RUN, THEN. RENUMBER will not renumber any reference that is part of a REMark.

RENUMBER commands may be put in BASIC programs, but on execution RENUMBER will return to BASIC command mode with the] prompt. RENUMBER is about 2K bytes long; it may take up to one minute to renumber or merge a 16K BASIC program. Remember the warning about not hitting Ctrl-Reset during this process.

The intended use for merge is to be able

Continued on pg. 34

FORTH WORLD

-Dudley Ackerman

New functions (commands) may be created using previously defined ones. These new definitions (words) may be used in further definitions.

Following is a definition we may usefully analyse.

```
: TEST-PROGRAM PAGE 5 0 VHTAB
  ." Input 2 numbers ( 0-65535)"
  QUERY INTERPRET CR
  ." Add or subtract?(a/s): " KEY  65 =
  IF + ELSE - THEN CR
  ." The answer is " U. CR ;
```

: - Start of a definition. (notice the space after the colon - all forth words need at least one space to separate them.

TEST-PROGRAM - This is the name of our new word. it may have up to 31 characters.

PAGE - clear the screen & place the cursor at the top left corner.

5 0 VHTAB - The phrase places the cursor at line 5 column 0. The range is 0-23 for lines; 0-39 for columns.

." - When the word this is in executes, the text after the ." is printed up to the next quote (") .

QUERY - Get characters up to a maximum of 80 or a carriage return.

INTERPRET - This is the heart of the forth system and is available to the user. It parses the text got by QUERY and tries to match these words with words in the dictionary. If a match is found the word is executed, if it is a valid number it is put on the stack. If neither, it is an error.

CR - Issue a carriage return.

KEY - Wait for a key to be pressed and put its ascii value (high bit off) on the stack.

65 - Put a 65 on the stack.

= - If the top two numbers are equal, take them off and leave a '1'; else take them off and leave a '0'.

IF ELSE THEN - If the top of the stack is non-zero execute words between IF and ELSE and continue after THEN. if it is zero, execute the words between ELSE and THEN. ELSE is optional. Without it, non-zero causes execution of words from IF to THEN ; a zero, causes execution to continue after THEN.

U. - Print the top of stack as an unsigned number. examples:
-1 prints as 65535
-2 prints as 65534
-32768 prints as 32768 etc.

; - Ends the definition.

Remember, forth words need at least one space between them, numbers can't have punctuation and a carriage return ends a line of input.

So we see our test word inputs two nu mbers and on reply to a prompt, adds or subtracts the two numbers and prints the result.

Next lesson, we may examine looping structures.

Our forth interest group thrives on the first & third Tuesday of each month. Call me at 626-6295 for information. And the Apple Core distributes our forth system and tutorial.

Forward with forth.

Announcing:

ReWrite
&
Information Manager

Two Exciting New Products

From

Systems Technology

SOFTWARE

See following two pages for details

Systems Technology
MICROCOMPUTER SOFTWARE

2270-I OAK GROVE ROAD
WALNUT CREEK, CA 94598
TEL: (415) 933-1134

Features of **ReWrite**

40/70/80 Column Upper/Lower Case Display. ReWrite allows the use of Apple II or Apple II Plus 40 column upper case mode and will capitalize through the use of the ESCape key. It allows use of the Apple //e 40 column upper/lower case with the SHIFT key. It supports the use of a SHIFT key modification on Apple II and Apple II Plus computers. It supports all 80-column cards used with the Apple II and Apple II Plus if the cards support Pascal (almost all do). It supports the Apple //e 80-column card and extended 80-column card. It provides an alternate 70 column high resolution graphic character set.

Editing. ReWrite supports: Automatic or manual search and replace; block move; block delete; block copy; append; screen print; left, right, center and fill justification; headers; footers; margins; boldface; italics; superscript; subscript; underlining; comments; and rapid cursor movement.

Printing. ReWrite supports bold (or shadow) print, italics, underlining, superscript, subscript, variable line spacing, page numbering with different styles, file linking, and allows special printer control codes. As delivered, ReWrite comes with printer control codes for over forty popular models of printers, both letter quality and dot matrix. Printer buffer spooling is supported on the Apple //e or the Apple II and Apple II Plus with a memory expansion card (16k or more).

Spellers. ReWrite supports the use of spelling checkers like Sensible Speller from Sensible Software.

Manual. ReWrite comes with a 35 page manual written in a friendly style.

Mail Merge. ReWrite supports mail merge capability using the data base INFORMATION MANAGER (also from Systems Technology).

Speed. ReWrite is written in 6502 Assembly Language and employs a modified DOS 3.3 operating system to assure fast operation.

Ordering Information. The price of ReWrite is $99. Mail and Phone orders are welcome. VISA, MasterCard, money order, certified check and cashier's checks will obtain immediate shipment. Personal checks must clear prior to order shipment.

Systems Technology
MICROCOMPUTER SOFTWARE

2270-I OAK GROVE ROAD
WALNUT CREEK, CA 94598
TEL: (415) 933-1134

Features of INFORMATION MANAGER

Information Manager is supplied on three diskettes; the PROGRAM diskette, the UTILITIES diskette and the SAMPLE DATA diskette. The back side of the UTILITIES diskette contains printer configuration data for 41 models of popular dot matrix printers. The first time you use **Information Manager** you will tell it what printer you have and it will customize itself to support your printer's features.

With the PROGRAM diskette you can: Define your own file structure (up to 30 fields), Add records, Change records, Delete records, List all or some records to screen or printer, Search for records (wildcards characters supported), Sort records, Move files to another disk, Format and print reports to screen, printer or disk, Format data diskettes (with added storage), and more. A screen dump feature is provided when listing or searching for records.

With the UTILITIES diskette you can: Change your file structure (your data files will be altered to the new format automatically), Append files, Copy data diskettes (or other unprotected diskettes), Check the space on any diskette, Obtain helpful information, Print your file structure, Print out your "raw" data, Recover from goofs, and automatically create a database containing all your normal DOS 3.3 diskettes.

With the SAMPLE DATA diskette you can: Breeze through the 25 page tutorial in the **Information Manager** manual, See examples of files, records and report formats, and learn the other program features.

Information Manager supports printers made by the following companies: Anadex, Apple, Axiom, C-Itoh, Centronics, Epson, Integral Data Systems, M.P.I., NEC, Okidata, Star Micronics and Transtar. Letter quality printers are supported in the "generic" mode.

The Manual provides, in addition to a 25 page tutorial, a 60 page Command Reference Guide which explains how to use each command on the Main Menu and the Utility Menu. An introduction is provided with general information. Seven appendicies are provided and include sample program outputs, error codes and recovery procedures, printer information, programmer's information, 80-column card usage (limited), a glossary, and helpful hints. The manual is indexed.

Equipment Required: Four versions of **Information Manager** are currently available: 1) Apple II/II+ with floppy drive, 2) Apple II/II+ with hard drive, 3) Apple //e with floppy drive, 4) Apple //e with hard drive. Apple II models require an Applesoft firmware card or a language card or memory expansion card in slot #0. The Apple II/II+ versions will run on the Apple /// under the emulation mode.

Ordering Information: The price of **Information Manager** is $195. Registered owners of THE INFORMER, THE INFORMER II or THE INFORMER]I[may upgrade to **Information Manager** for $100 by sending their original INFORMER diskette with their order. Mail and phone orders are welcome. VISA, MasterCard, money order, certified check and cashier's check orders will obtain immediate shipment. Personal checks must clear prior to order shipment.

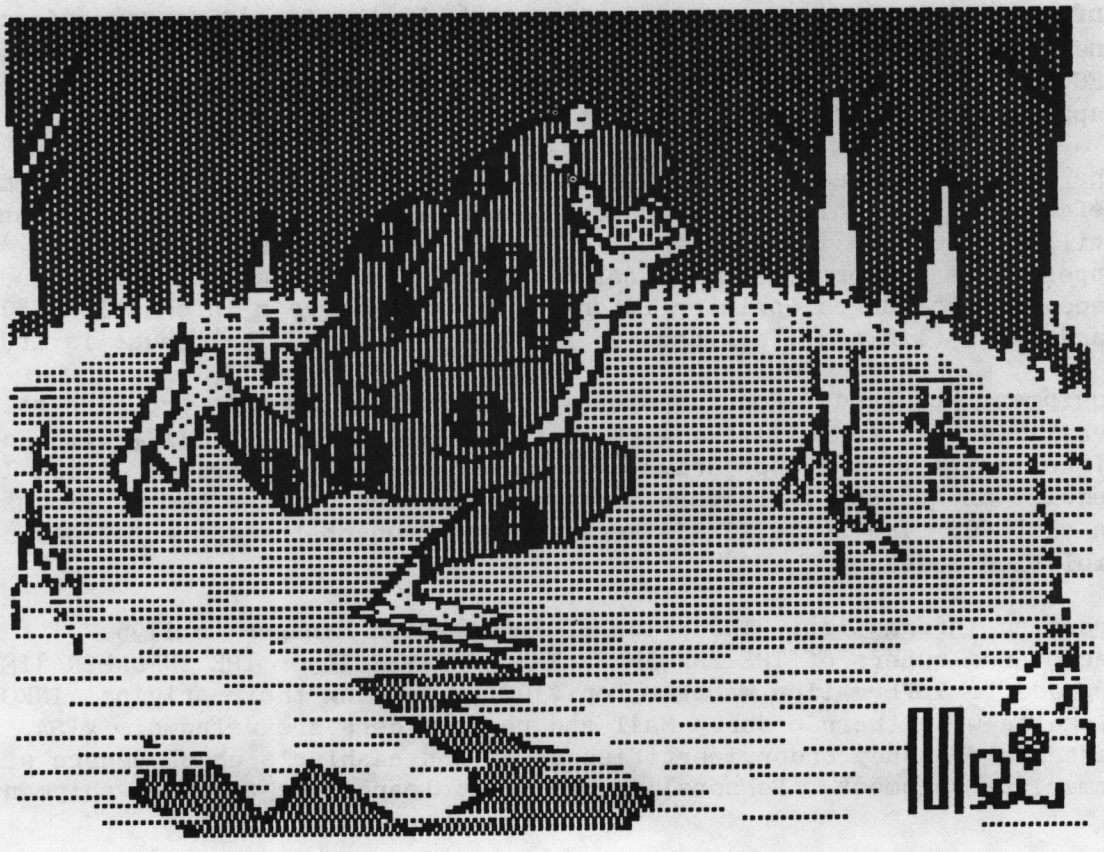
Quickie: Center Formatting

-Max J. Nareff

String centering is based on the subtraction of the length of the string from the total screen width, and subsequently, division of the result by 2. This number can be used as the column variable in the GOTOXY (X, Y) procedure.

If the field-width indicator (String:X) is used, the length of the string is added to the calculated row variable. The following "shorty" illustrates the mechanism. Additional examples may be found in the Sept/Oct.1982 issue of the CIDER PRESS.

```
PROGRAM CENTERDEMO;
VAR
   S : STRING;
   LINE : INTEGER;

PROCEDURE CLEARLINE (X, Y : INTEGER);
BEGIN
   GOTOXY (X, Y);
   (* Clear to end of line *)
   WRITE (CHR (29))
END;

PROCEDURE CENTER (ROW:INTEGER;S:STRING);
CONST
   XMAX = 79; YMAX = 23;
VAR
   I : INTEGER;
BEGIN
   IF (ROW > YMAX) THEN LINE := YMAX;
   I := (XMAX - LENGTH (S)) DIV 2;
   IF (I < 0) THEN I := 0
   ELSE
    IF (I > (XMAX DIV 2))
          THEN I := XMAX DIV 2;
   CLEARLINE (0, ROW);
   GOTOXY (I, ROW);
   WRITELN (S);
   WRITELN (S:I + LENGTH (S))
       (* Alternate format mode *)
END;

BEGIN  (* MAIN *)
   PAGE (OUTPUT);
   WRITELN ('Enter string
          to be centered.');
   READLN (S);
   WRITELN ('On what line?');
   READLN (LINE);
   CENTER (LINE, S)
END.
```



Avoiding Array Disarrays

-Max J. Nareff

The commonest error encountered in using arrays is the range error, where the dimensions of the array are exceeded. Experienced programmers carefully select precise or adequate dimensions to match the actual or potential number of values (contents, elements, components) to be stored in the array.

How then can one cope with a single program that may be required to manage arrays of varying sizes? Two approaches are offered.

#1: Assume all appropriate variables have been declared and defined, and that the array is adequately dimensioned. First read the data from the file by means of a "While" loop and asssign it to indexed array variables through a counter.

```
Then USE THE COUNTER to control
subsequent array operations.

RESET (F, 'ONESIZE.TEXT');
K := 1;
READLN (F, STR);
WHILE NOT EOF (F) DO
   BEGIN
      ALPHA[K := STR;
      K := K + 1;
      READLN (F, STR)
   END; (* WHILE *)

FOR I := 1 TO K DO
   BEGIN
(* Manipulate individual array elements,
sort, etc. *)
                  ----------
                  ----------
   END; (* FOR *)
```

#2: Involves coding the different sized file names in such a way that the using program will determine the size of each data file, and thus have a value for use in the re- quired loop. PLACE THE SIZE OF THE DATA FILE IN THE FILE NAME. The following FUNCTION peels the number from the file name for use in a loop.

```
FILENAME := 'MYSIZEIS50.TEXT';

TEMPNAME := FILENAME;
FUNCTION FILESIZE
(VAR NAME:STRING):INTEGER;
VAR I, DIGITS : INTEGER;
BEGIN
   DIGITS := 0;
(* Remove Vol.# in case used in filename
prompt *)
   IF POS (':', NAME) <> 0 THEN
   NAME := COPY(NAME,POS(':',NAME)+1,
   LENGTH(NAME)-POS(':',NAME))
   FOR I := 1 TO LENGTH (NAME) DO
      DIGITS := DIGITS*10 + ORD(NAME[I])-
                ORD('0');
   FILESIZE := DIGITS
END; (* FILESIZE *)

   SIZE := FILESIZE (TEMPNAME);

   FOR I := 1 TO SIZE DO
      BEGIN
         READLN (F, STR);
                ALPHA[I := STR;
         .......
         .......
      END; (* FOR *)

FOR J := 1 TO SIZE DO
   BEGIN
      ALPHA[J := .........;
      .........
   END; (* FOR J *)
```

Use of these simple methods makes it possible to manage arrays of varying sizes and avoid range errors.

DOT MATRIX PRINTER POWER FROM APPLE WRITER] [- //e

Glossary One

Provides the easiest, most complete access of Epson, Gemini and Apple Dot Matrix print features and characters.

only 21.95

GlossaryOne with NulFix™

NulFix allows Epson MX and Gemini users to access features (e.g. superscripts) previously unobtainable from Apple Writer //e. (NulFix not needed with Apple Writer] [versions).

only 27.95

Phone orders and inquiries invited!
Or send check or money order to:

MIJO CO.
1 Jo-Lin Ct. El Sobrante, CA 94803
(415) 655-2158

Specify Apple Writer] [or //e. Epson (MX, FX, RX), Gemini or Gemini-X or Apple Dot Matrix

Interrupts on the Apple II

-Max J. Nareff

Users of the Apple III may have noticed a recent increase, albeit belatedly, in the number of pertinent articles in the magazine literature. Apple Orchard, Softalk and InCider have been the pacesetters. In reference to Pascal applications, the magic assembly - language contributions of Dr. J. Jepson, in Softalk, stand out.

Another seminal article, a must read, is Al Evans contribution in Byte Aug.83, "HELP In APPLE III Pascal", which presents methods of interrupting running programs.

Mr. Evans, an obviously sophistocated programmer, encourages the use of his imaginative efforts by his public domain assignments. Readers of this column may recall his striking arrow-controlled menu program which is in the clubs Pascal library.

This two-part series is based on his Byte article and presents an adaptation and modification of the demonstration for use on the APPLE II.

Obviously, some of the advanced capabilities resident in the APPLE III can not be translated for the APPLE II, and have been omitted. Nevertheless, the basic procedures are noteworthy, idea-provoking and readily adaptable.

Interrupt: Temporarily stops an operating program, executes some other code and then "resumes operation at the point of interruption". An interrupt is usually caused by a signal from an external source, or by clock or communication cards, and is often used to allow two programs to operate "at the same time".

From a practical standpoint, the nearest thing to an interrupt is the KEYPRESS function in APPLE Pascal, residing in the Applestuff unit. It was provided because the APPLE keyboard is not interrupt-driven.

Interrupt action is simulated by KEYPRESS, by polling the keyboard frequently and storing the result in a buffer. If, as a result of a keypress, the buffer contains a character, the function returns TRUE.

However, it does not read the character from the keyboard, indicating only that something is there "waiting to be read". In its commonest use, KEYPRESS simulates the interrupt process by permitting the program to digress and to respond to the instructions which follow the keypress call.

Supposing, that instead of burdening a program with these conditional routings, one could interrupt the program, AT ANY TIME, call in on-line instructions, etc from an outside source, such as a disk drive, and then RESUME the program at its point of interruption.

Evans points out that such a process could be used to provide help (1) At the menu level - for exampe to explain the choices; (2) During keyboard input - to obtain information about what and how to enter; (3) During a continuous process to interrupt data, etc being displayed on the screen, for explanation, etc.

To obtain these interesting effects, Evans utilizes keyboard sensing and several unique procedures, without recourse to assembly-language programming.

The original complete demo program, for the APPLE III and its APPLE II adaptation, will appear in a forthcoming DOM, but the Byte article itself merits study.

Due to space limitations, the modified program is presented in two parts. The first will contain the program skeleton and an executable section; the second will include the remainder, to be added to part one for completing the program.

The user will first need to enter the E(ditor and prepare up to 22 lines of a sample instruction page. Evans uses part of his subject code for this, but it seems simpler to go the sample route. Save the "instruction sheet" as "INFOSHEET1".

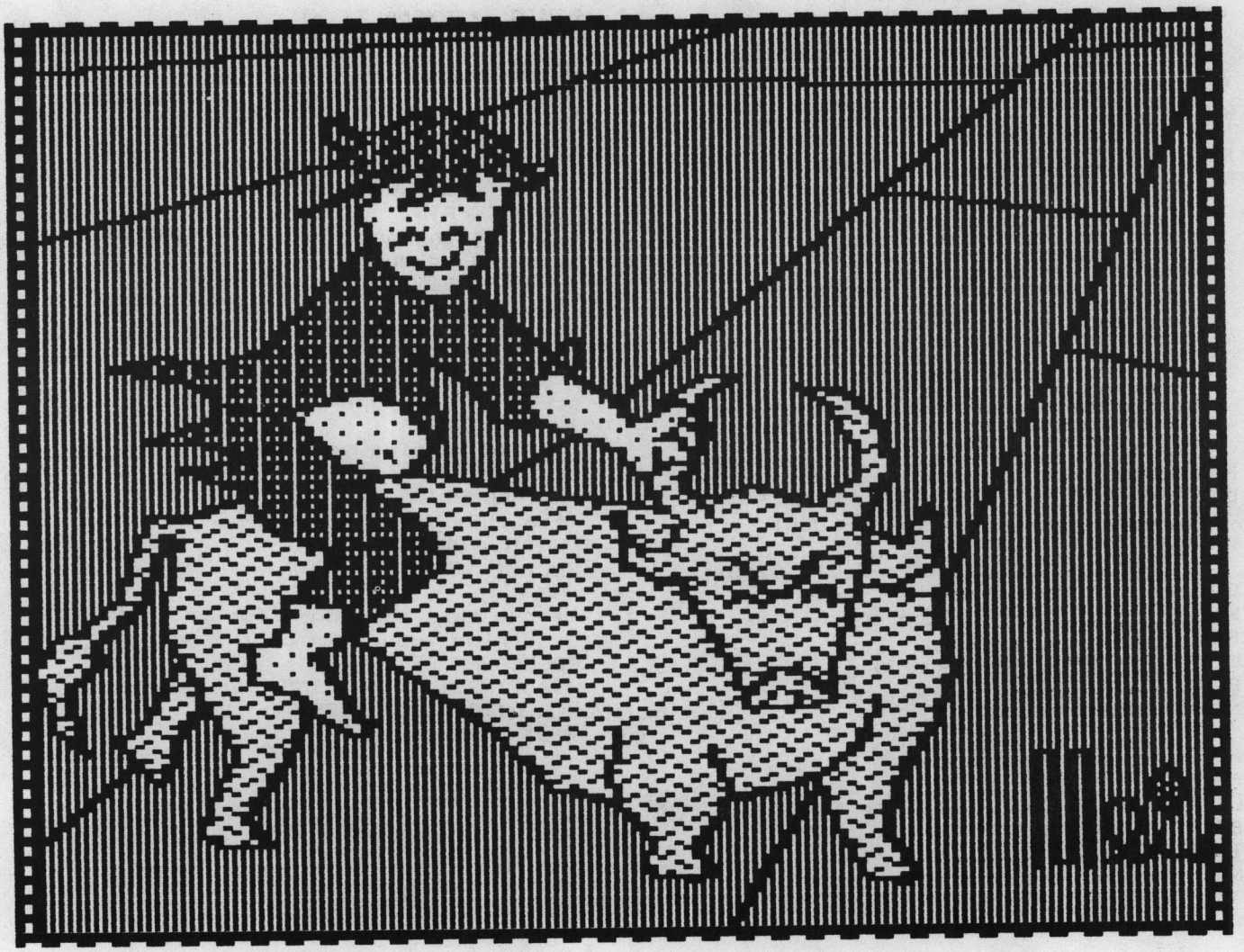
Enter the F(iler and change the saved filename to just "INFOSHEET1", without the ".TEXT". This is because Evans, desiring an instaneous read of the disk file, elected to use the BLOCKREAD function, which requires an untyped file. Enter the following program.

```
(*$S+*)

(* A demonstration of a HELP program.
Derived from Al Evans's article *) (*
BYTE Aug.83, "Help in Apple III Pascal".
Changes made to adapt for *) (* the
Apple II. Prepared by Max J. Nareff 1/84
*)
```

```
PROGRAM HELPDEMO;
USES APPLESTUFF;  (* For Keypress
                     function *)
CONST
 MENU1 = 1; MENU2 = 2; EX1 = 1; EX2 = 2;
TYPE
 Byte = 0..255;
 Buffer2K = Packed Array[0..2048 of Byte;
 MenuMode = (NORMALMODE, HELPMODE);
VAR
 LEFTARROW, ESCAPE, HELPKEY, BEEP : CHAR;
 INPUTSET : SET of CHAR;

PROCEDURE BACKSPACE;
BEGIN  WRITE (CHR(8))  END;

PROCEDURE HELP (WHERE : INTEGER);
CONST
 STARTBLOCK = 2;(*To omit header blocks*)
 PREFIX = '#5:INFOSHEET';
VAR
 FRAME : Buffer2K;BLOCKCOUNT : INTEGER;
 FILENAME : STRING;HELPFILE : FILE;
 KEY : CHAR;CHTOSTR : STRING[1;
BEGIN
 PAGE (OUTPUT);
 STR (WHERE, FILENAME);
 FILE_NAME := CONCAT (PREFIX, FILENAME);
 (*$I-*)
 RESET (HELPFILE, FILENAME);
 (*$I+*)    (* The IO check can be omitted
   if INFOSHEET will always be on line.
   Omit code from here to UNTIL... *)
 IF IORESULT <> 0 THEN
    REPEAT
       WRITELN
(BEEP, 'Put Program Disk online and press
<RETURN>, or <ESCAPE> to Exit: ');
       READ (KEY);
       IF KEY = ESCAPE THEN
          BEGIN
             PAGE (OUTPUT);
             EXIT (HELP)
          END
       ELSE
            (*$I-*)
          RESET (HELPFILE, FILENAME)
            (*$I+*)
    UNTIL IORESULT = 0;
 BLOCKCOUNT :=
 BLOCKREAD
  (HELPFILE, FRAME, 4, STARTBLOCK);
 UNITWRITE
  (1, FRAME, (BLOCKCOUNT * 512), 0, 0);
 GOTOXY (15, 23);
 WRITE ('Press any key to proceed');
 READ (KEY);
 CLOSE (HELPFILE);
 PAGE (OUTPUT)
END; (* HELP *)

PROCEDURE INITVARIABLES;
BEGIN
 LEFTARROW := CHR(8);BEEP := CHR(7);
 ESCAPE := CHR(27); HELPKEY := '?';
 INPUTSET := [CHR(32)..CHR(126);
END; (* INITVARS *)
```

```
PROCEDURE EXAMPLE1;
(* An example of HELP during a continuous
screen display - HELP would normally be
used in this instance to provide info on
on the format and/or meaning of the
display*)
VAR
 FOREVER : BOOLEAN;

 PROCEDURE HANDLEKEY;
 VAR KEY : CHAR;
 BEGIN
    READ (KEYBOARD, KEY);
    IF KEY = HELPKEY THEN HELP(EX1);
    IF KEY = ESCAPE THEN EXIT(EXAMPLE1)
 END; (* HANDLE_KEY *)

 PROCEDURE SLOWPRINT (LINE:STRING);
(* This procedure given as an extreme ex.
of interruption during display-process can
be interrupted after each char appears.*)
 VAR  CHARCOUNT : INTEGER;
 BEGIN
    FOR CHARCOUNT := 1 TO LENGTH (LINE) DO
       BEGIN
         WRITE (LINE[CHARCOUNT);
         IF KEYPRESS THEN HANDLEKEY
       END
 END; (* SLOWPRINT *)

BEGIN (* EXAMPLE1 *)

 FOREVER := FALSE;
 PAGE (OUTPUT);
 REPEAT
    SLOWPRINT
('This is an example of how HELP
can be provided during ');
    SLOWPRINT
('an ongoing process. This text will
be repeated until ');
    SLOWPRINT
('<ESCAPE> is pushed. During display,
pressing the HELP ');
    SLOWPRINT
('KEY -> <?> will cause your help
screen to be shown. ');
    SLOWPRINT
('After reading your help screen,
the program will return');
    SLOWPRINT
('to this display.')
 UNTIL FOREVER
END; (* EXAMPLE1 *)

(* Use this procedure for interruptible
   string input. In part 2 *)
PROCEDURE GETSTRING
(VAR INPUTLINE:STRING;
 CALLINGPROC:INTEGER);
BEGIN  END;
```

```
(* In part 2 of this article *)
PROCEDURE EXAMPLE2;
BEGIN  END;
PROCEDURE MENU(MODE : MENUMODE);
VAR
 CHOICE : CHAR;
BEGIN
 REPEAT
    PAGE (OUTPUT);
    IF MODE = NORMALMODE THEN
       BEGIN
        GOTOXY (0, 20); WRITELN
        ('Use the <?> for help!')
       END;
    GOTOXY (0, 4);
    WRITELN
(' ':5,'Demonstrations available:');
 WRITELN; WRITELN
(' ':5,'[1:HELP during continuous
    output');
    WRITELN
(' ':5,'[2:HELP during input from
    keyboard');
    WRITELN;
    IF MODE = HELPMODE THEN
       BEGIN
         WRITE (' ':5);
         WRITELN (' Now in HELP mode.');
         WRITELN
       END;
    WRITELN; WRITELN
(' ':5,'Enter option or
   <ESCAPE> TO quit');
    READ (CHOICE);
    IF MODE = NORMALMODE THEN
       CASE CHOICE OF
         '1' : EXAMPLE1;
         '2' : EXAMPLE2
       END (* CASE *)
    ELSE
(*Should the user want to see help sheet*)
       CASE CHOICE OF
         '1' : HELP (MENU1);
         '2' : HELP (MENU2)
       END; (* CASE *)
    IF (MODE = NORMALMODE) AND
 (CHOICE = HELPKEY) THEN MENU (HELPMODE)
 UNTIL CHOICE = ESCAPE
END; (* MENU *)

BEGIN (* Main Program *)
 INITVARIABLES;
 MENU (NORMALMODE);
 PAGE (OUTPUT)
END.
```



Easy Pascal I/O Routines

Copyright 1984 by Michael Fischer

Part 3 of a Series

[This article uses CAPITALS to identify procedures, functions and variables created by the program and "Quotes" to indicate intrinsic procedures, functions and variables.]

The LONGIODEMO program consists of parts of the INTIODEMO program presented in part one of this series and the additional text in listings 1 and 2.

From the command line press "E" (editor) and return for a new work file. Press "C" (copy) and "F" (from file). Give the editor the file name which contains your INTIODEMO text file. Include the volume name. You need not add the suffix ".Text" because the editor will do so.

If you have one drive, be sure the disk with the INTIODEMO text file is in the drive before pressing return. If you removed your boot disk from drive 1 (volume 4), be sure to replace it after copying.

Make the following changes in the copied file:

1. Delete the following global variables (listed at the beginning of the program): INTVALUE, LOWIRANGE and HIGHIRANGE.

2. Add the following global variables, of type LONG: HIGHLRANGE, LOWLRANGE and LONGVALUE.

3. For the sake of clarity you should change the program name to something like LONGIODEMO, although it will compile and run if you don't change it.

4. Delete the code for the functions IVERT, PIVERT, and RIVERT.

5. Change the line in PROCEDURE INTRO reading "Writeln ('string inputs converted to integer.');" by adding the word "long" before the word "integers."

6. Globally replace HIGHIRANGE with HIGHLRANGE, LOWIRANGE with LOWLRANGE, INTVALUE with LONGVALUE, and PIVERT with PLVERT. To globally change press "J" (jump), "B" (beginning), "/", "R", "/Highirange/Highlrange/" and each occurance of HIGHIRANGE will be replaced with HIGHLRANGE. Repeat for the other variables.

7. Delete from "Begin (*Main*)" to the end of the text file.

Move the cursor to the beginning of PROCEDURE INTRO, press "I" for Insert, and then enter the text in listing 1. Move the cursor to the end of the file ("J" - jump; "E" - end), press "I" for Insert, and then enter the text in listing 2.

Long Integer Routines

The long integer routines are quite simple. Each of the procedures merely takes the entered value, as a string, and converts it to a long integer using the same process as was used in converting an entry to an integer. Indeed the same procedures are used. Only conversion stops before the number is truncated from a long integer into an integer.

The main long integer processes are the ones that will actually be called by your program. While the integer and real processes are functions, the long integer routines are procedures. Pascal requires that functions be restricted to scalars, subranges or pointers and long integers are none of these. Therefore the actual value of the long integer is passed by reference (as a "Var" parameter).

The naming convention, which by now should be quite familiar, uses "l" (for long integer) as the root and a suffix of "vert" (for convert). Finally each name may have a prefix of up to two letters -- a "P" if a prompt message is desired before the input, and "R" if the number must be within a specified range, a "Pr" if both options are wanted, and no prefix if neither option is wanted.

Parameters passed to the procedure consist of 4 integers -- an X and Y screen location for the prompt/data input, and an X and Y screen location for any error message. Additional parameters are required if the "P" option (the prompt line -- a string) or the "R" option (the low and high ends of the range as long integers) are selected. The "P" option parameters preceed the "R" option parameters if both options are selected. Finally the variable which will hold the converted long integer must be the last in the list of parameters.

The routines are called by your program with a statement such as:

```
Prlvert (0,0,0,23,'Number: ',3000,
  600000,Longone);
```

This would accept data beginning with a prompt message "Number: " at screen location 0,0 (upper left corner) and print any error message at screen location 0,23 (lower left corner). When a proper long integer between 3,000 and 600,000 has been entered its value will be placed in Longone (or whatever name you use).

Long integer input error messages are:

"Misplaced + sign." -- The + sign used was not the first character.

"Misplaced - sign." -- The - sign used was not the first character.

"Non-number found." -- A non-number was found.

"Too many digits." -- The long integer contained more than 36 digits.

"Value too low; must be ### or more." -- The value entered was less than the minimum specified for the range.

"Value too high; must be ### or less." -- The value entered was greater than the maximum specified for the range.

"No number entered." -- No number remains after removing leading blanks.

Each of the main procedures begins by clearing the entry line from its starting position using the CLEARLINE porcedure. Next the cursor is placed at the input location using the "Gotoxy" procedure, any prompt written, and data input accepted as a string. Next INTCONVERT is called. If there is an error (OKAY=false), the error message line is cleared and the error message is written. If range checking is specified the range is checked and an error message is written if the number (LONGINT) is outside the specified range (less than LOW or greater than HIGH). Each routine repeats until there is no error (OKAY=true) and, if range checking was specified, LONGINT <= HIGH and >= LOW. When there is no error the procedure erases any error message and returns to the calling routine.

The main program in listing 1 is just a simple demonstration program using the entry routines. In actual use you could replace it with your own main program and any of your own procedures and functions. Including the coding for all the procedures and functions add considerable size to program text and code files. A solution, using "Units" and "Libraries", will be presented in a later article in this series.

Listing 1

```
Procedure Lvert (X,Y,Errx,Erry:Integer;
  Var Longint:Long);
  Const
    Longlength = 36;
  Var
    Okay                    : Boolean;
    Entry,
    Errmessage              : String;
  Begin
    Repeat
      Clearline (X,Y);
      Gotoxy (X,Y);
(*Get entry.*)
      Readln (Entry);
(*Convert it.*)
      Intconvert (Entry,Longlength,Longint
        Okay,Errmessage);
      If (Not Okay)
        Then
(*Print error message.*)
          Begin
            Clearline (Errx,Erry);
            Gotoxy (Errx,Erry);
            Write (Errmessage)
          End
    Until Okay;
(*Erase error message line.*)
    Clearline (Errx,Erry);
  End;

Procedure Plvert (X,Y,Errx,Erry:Integer;
  Prompt:String; Var Longint:Long);
  Const
    Longlength = 36;
  Var
    Okay                    : Boolean;
    Entry,
    Errmessage              : String;
  Begin
    Repeat
      Clearline (X,Y);
      Gotoxy (X,Y);
(*Prompt for entry.*)
      Write (Prompt);
      Readln (Entry);
(*Convert it.*)
      Intconvert (Entry,Longlength,Longint
        Okay,Errmessage);
      If (Not Okay)
        Then
(*Print error message.*)
          Begin

            Clearline (Errx,Erry);
            Gotoxy (Errx,Erry);
            Write (Errmessage)
          End
    Until Okay;
(*Erase error message line.*)
    Clearline (Errx,Erry);
  End;

Procedure Rlvert (X,Y,Errx,Erry:Integer;
  Low,High:Long; Var Longint:Long);
  Const
    Longlength = 36;
```

```
  Var
    Okay                    : Boolean;
    Entry,
    Errmessage              : String;
  Begin
    Repeat
      Clearline (X,Y);
      Gotoxy (X,Y);
(*Get entry.*)
      Readln (Entry);
(*Convert it.*)
      Intconvert (Entry,Longlength,Longint
        Okay,Errmessage);
      If (Not Okay)
        Then
(*Print error message.*)
          Begin
            Clearline (Errx,Erry);
            Gotoxy (Errx,Erry);
            Write (Errmessage)
          End
(*Check range limits.*)
        If (Okay) and (Longint < Low)
          Then
            Begin
              Clearline (Errx,Erry);
              Gotoxy (Errx,Erry);
              Write ('Value too low; ',
                'must be ',Low,
                ' or more.');
            End;
        If (Okay) and (Longint > High)
          Then
            Begin
              Clearline (Errx,Erry);
              Gotoxy (Errx,Erry);
              Write ('Value too high; ',
                'must be ',High,
                ' or less.');
            End
    Until (Okay) and (Longint <= High)
      and (Longint >= Low);

(*Erase error message line.*)
    Clearline (Errx,Erry);
  End;

Procedure Prlvert (X,Y,Errx,Erry:Integer;
  Prompt:String;Low,High:Long; Var
  Longint:Long);
  Const
    Longlength = 36;
  Var
    Okay                    : Boolean;
    Entry,
    Errmessage              : String;
  Begin
    Repeat
      Clearline (X,Y);
      Gotoxy (X,Y);
(*Prompt for entry.*)
      Write (Prompt);
      Readln (Entry);
(*Convert it.*)
      Intconvert (Entry,Longlength,Longint
        Okay,Errmessage);
      If (Not Okay)
        Then
(*Print error message.*)
          Begin
            Clearline (Errx,Erry);
            Gotoxy (Errx,Erry);
            Write (Errmessage)
          End
(*Check range limits.*)
        If (Okay) and (Longint < Low)
          Then
            Begin
              Clearline (Errx,Erry);
              Gotoxy (Errx,Erry);
              Write ('Value too low; ',
                'must be ',Low,
                ' or more.');
            End;
        If (Okay) and (Longint > High)
          Then
            Begin
              Clearline (Errx,Erry);
              Gotoxy (Errx,Erry);
              Write ('Value too high; ',
                'must be ',High,
                ' or less.');
            End
    Until (Okay) and (Longint <= High)
      and (Longint >= Low);
(*Erase error message line.*)
    Clearline (Errx,Erry);
  End;
```

If you are interested in a
C language
interest group, contact:

Bryan Costales

641-7929

```
#########################################
#                                       #
#          AUTHORS WANTED               #
#                                       #
#  Share your experiences, Apple        #
#  programming hints,pascal tips,       #
#  software reviews. Send your          #
#  material to the Cider Press          #
#  Editor at the Clubhouse......        #
#                                       #
#########################################
```

Filemanager/Machine Language Interface

-Bryan Costales

Part 2 of a Series

(c) B. Costales 1983

ON OOPS AND FREE STUFF

Since my last writing, I have been mortified to discover that the break vector does not work with Integer BASIC nor the Non-Autostart monitor rom. I made this discovery quite by accident, when attempting to run FMLI on a friend's computer. His Apple is a pre-revision 7 model, with Applesoft on a firmware card! Under the Old monitor rom, when you attemt to execute a BRK instruction, WHAM! You drop straight into the system monitor, all register exposed. -- Obviously this little oversight makes FMLI much less universally useful than I had planned. After much self recrimination, I have re-thought my approach to system interfacing, and returned to the land of the page-three vector.

I am hoping no one was silly enough to type in the program listing which appeared last issue. If you did, however, take solice in that only a handful of lines have been changed. Three cheers for modular programming. -- I heartily recommend that you do not enter the listing as it appears and will appear. With the conclusion of these articles next issue, I will make available thru the San Francisco Apple Core, both the object and source code on DOS 3.3 diskette. The value of the listings lies in their documentation of the program.

NOTICE is Hereby Given that

The "Annual Meeting" of the

San Francisco Apple Core, Inc.

will be held on

Saturday, April 14th

Bldg. C Fort Mason Center

at 10:00 A.M.

Election of Officers to the Board of Directors

for the coming year
will be held at this meeting.

THE FILENAME

Under DOS 3.3, filenames may legally begin with any character above the ascii value "@". They may include any character excepting the comma and carriage return. As a programmer interacting with the real world, this freedom can become a real annoyance. Suppose you create a progam "foo" which prompts for a filename, then reads in that file. It is not foolish to anticipate a user entering:

ENTER FILENAME: MY LETTER(2) ,S5 ,D2

Under FMLI, the chore of stripping the slot and drive specifications, and of parsing out the filename and padding it with spaces, is neatly built in. To open the file, simply pass as one of the arguements to FOPEN, the address of the raw filename. $200, or the standard input buffer here. FMLI does all the work. A call to FGETNAME will return to you the address of a buffer containg the pure, stripped, padded filename.

The job of range checking is left to you. Since some hard disks patch RWTS to utilize unusual slot and drive specifications, you must call FGETSDV to see that something legal was entered for your system. -- FMLI does not check for ",A ,L ,R or ,B" specifications. You must pre-screen the input filename for these paramaters if you expect them. FMLI won't bomb if given the unexpected, it merely ignores what it doesn't understand. As an added bonus, FMLI allows a filename to begin with and contain absolutely anything, excepting only the lowly comma.

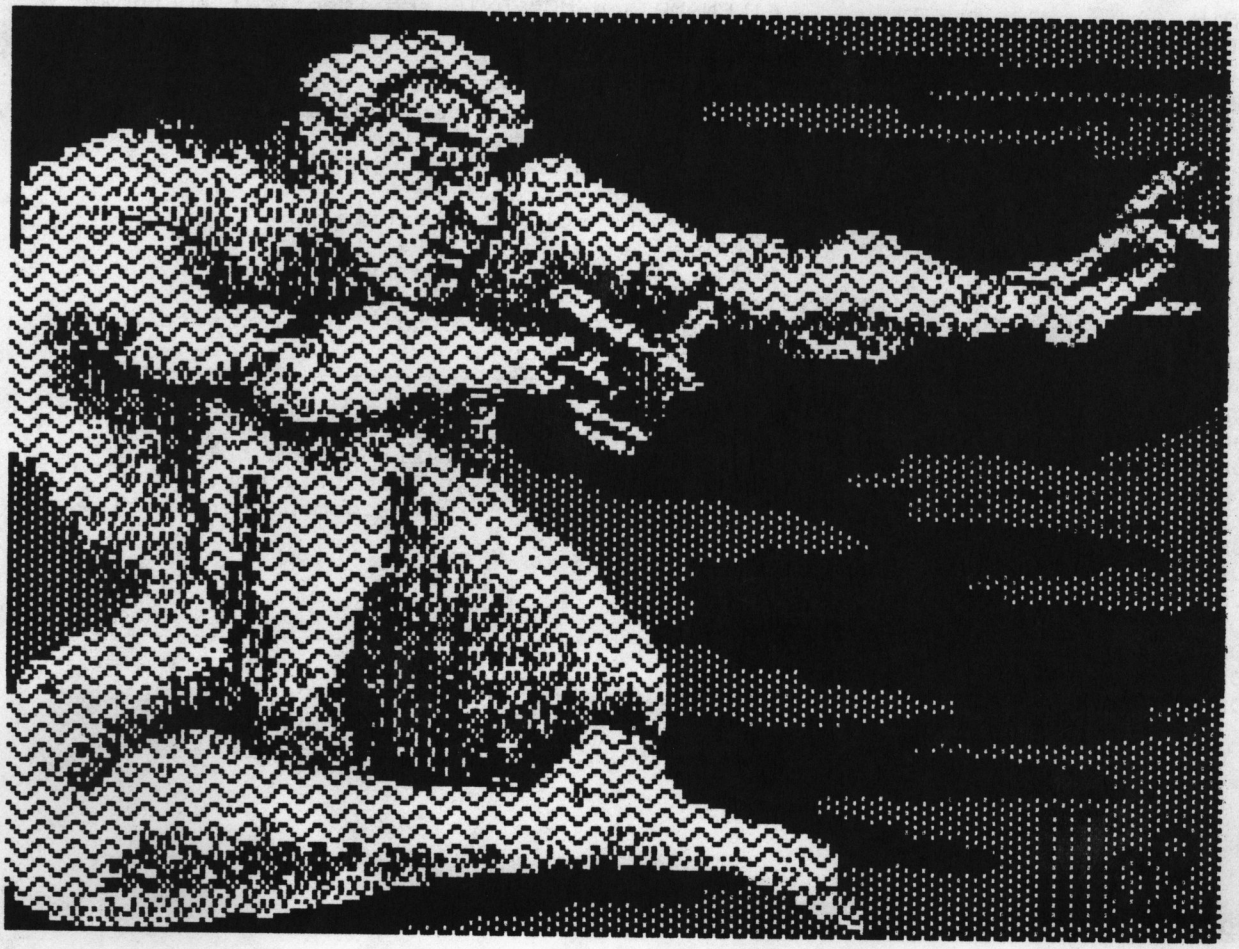
ON EXISTENCE

FMLI draws a hazy line between opening an existing file, and creating a new file. The DOS Filemanager is the culprit here. Ideally, should you wish to create a new file, the operating system should complain if that file already exists. Under FMLI, creating a new file will not complain at all if the file already exists. Even if it is the wrong file type. This is not a bug. The onus is on the programmer to see that what he got is what he wants. The proper sequence to create a new file, where that file must not already exist, is:

1) Open the file with BFOPEN. If an error (file not found), skip to step 3.

2) Close the file with FCLOSE. Delete the file with first FCHMOD (to unlock it) then FUNLINK to delete it.

3) Create the file with FCREATE.



```
0B10                ;
0B10                ;-----------------------------------------------------------
0B10                BFOPEN:
0B10                ; Calling format:  JSR CALLFMLI
0B10                ;                  BYT FOPEN
0B10                ;                  ADR FILENAME
0B10                ;                  ADR MODEBYTE
0B10                ; Where:
0B10                ;        FILENAME is any zero or carriage-return
0B10                ;         terminated string. It may include comma
0B10                ;         delimited specifications for slot, drive,
0B10                ;         and volume.
0B10                ;        MODEBYTE is $00 for "read only", $01 for
0B10                ;         "write only", $02 for "read/write".
0B10                ; A-reg returns the "file descriptor" for the opened
0B10                ;  file. Errors return the carry flag set, and the
0B10                ;  a-reg containing the error number.
0B10                ; Call bftype to find the "type" of the open file.
0B10                ;-----------------------------------------------------------
0B10 A9 01                  LDA #$01
0B12 48                     PHA                  ; save on stack
0B13 20 E0 0B               JSR FALLOC           ; allocate a buffer
0B16 90 0E                  BCC COMOPEN          ; exit if error.
0B18 4C 9D 0B               JMP AERR2            ;Use common open routine
0B1B                ;
0B1B                ;-----------------------------------------------------------
0B1B                BFCREATE:
0B1B                ; Calling format:  JSR CALLFMLI
0B1B                ;                  BYT FCREATE
0B1B                ;                  ADR FILENAME
0B1B                ;                  ADR MODEBYTE
0B1B                ;                  ADR TYPEBYTE
0B1B                ; Where:
0B1B                ;        FILENAME and MODEBYTE are identical to those
0B1B                ;         for bfopen.
0B1B                ;        TYPEBYTE is:
0B1B                ;          $00 = "T" text file
0B1B                ;          $01 = "I" integer basic file
0B1B                ;          $02 = "A" applesoft file
0B1B                ;          $04 = "B" binary file
0B1B                ;          $08 = "S" system file
0B1B                ;          $10 = "R" relocatable file
0B1B                ;          $20 = "a" archive file
0B1B                ;          $40 = "L" lisa source file
0B1B                ; Otherwise,
0B1B                ;  identical to BFOPEN, excepting it will create a
0B1B                ;  file if it can't be found.
0B1B                ;-----------------------------------------------------------
0B1B A9 00                  LDA #$00
0B1D 48                     PHA                  ; save on stack
0B1E 20 E0 0B               JSR FALLOC           ; allocate a buffer
0B21 90 03                  BCC COMOPEN          ; exit if error.
0B23 4C 97 0B               JMP AERR3            ;Use common open routine
0B26                ;
0B26                        .IF PAGEON
0B26                        PAG
0B26                        .FI
0B26                ;-----------------------------------------------------------
0B26                COMOPEN:
0B26                ; Common open routine for both fopen and fcreate.
```

```
0B26                    68  ; Whether or not a file is created is determined
0B26                    69  ; by value on top of stack. $00 error if not found.
0B26                    70  ;-----------------------------------------------------------
0B26 20 F8 0A           71          JSR PULLADR         ;parse the filename.
0B29 20 B6 0C           72          JSR PARSENAM
0B2C 20 F8 0A           73          JSR PULLADR         ;save the mode byte.
0B2F 85 07              74          STA FRWSTAT
0B31 68                 75          PLA                 ;Is there a type byte?
0B32 48                 76          PHA
0B33 C9 00              77          CMP #$00
0B35 D0 05              78          BNE >1
0B37 20 F8 0A           79          JSR PULLADR         ; yes, place in param.
0B3A 85 03              80          STA FCURTYPE
0B3C 20 AB 0C           81  ^1      JSR STATSAVE
0B3F 20 94 0C           82          JSR SWAPUP          ;Copy up paramaters
0B42 20 C4 0B           83          JSR ZEROREC         ; and set record length
0B45 A9 01              84          LDA #$01            ; to zero.
0B47 8D BB B5           85          STA FMOPCODE        ;Opcode = 1 for open.
0B4A 68                 86          PLA
0B4B 48                 87          PHA
0B4C AA                 88          TAX
0B4D 20 FD AA           89          JSR CALLFM          ;Call the filemanager.
0B50 68                 90          PLA                 ;Create flag to x-reg.
0B51 AA                 91          TAX
0B52 90 0E              92          BCC >3              ;Carry clear no errors.
0B54 AD C5 B5           93  ^1      LDA FMRETURN        ;Error: get error number.
0B57 C9 05              94          CMP #$05            ;  file not found?
0B59 D0 04              95          BNE >2              ;  different error exit.
0B5B E0 00              96          CPX #$00            ;  okay to create?
0B5D F0 03              97          BEQ >3              ;  yes, continue.
0B5F 4C B1 0B           98  ^2      JMP MAPFMERR
0B62 AD C2 B5           99  ^3      LDA FMTYPE          ;Save actual type found.
0B65 85 03             100          STA FCURTYPE
0B67 20 94 0C          101          JSR SWAPUP          ;No error: copy up
0B6A A9 0A             102          LDA #$0A            ; paramaters and zero
0B6C 8D BB B5          103          STA FMOPCODE        ; length and offset.
0B6F 20 C4 0B          104          JSR ZEROREC         ; opcode=$0A position
0B72 20 CD 0B          105          JSR ZEROOFF         ; to start of file.
0B75 20 FD AA          106          JSR CALLFM          ;Call filemanager.
0B78 B0 E5             107          BCS <2              ;Carry set, exit error.
0B7A 20 AB 0C          108          JSR STATSAVE
0B7D AD 05 0A          109          LDA HEAPADR         ;Success!
0B80 85 FC             110          STA VP              ;Mark the buffer as
0B82 AD 06 0A          111          LDA HEAPADR+$01     ; in-use.
0B85 85 FD             112          STA VP+$01
0B87 A0 00             113          LDY #$00
0B89 B1 FC             114          LDA (VP),Y
0B8B A8                115          TAY
0B8C A9 01             116          LDA #$01
0B8E 91 FC             117          STA (VP),Y
0B90 A5 06             118          LDA FCURFD          ;Set a-reg to return the
0B92 85 F5             119          STA ASAVE           ; file descriptor.
0B94 4C DD 0D          120          JMP COMCLCEX
0B97                   121  ;
0B97                   122          .IF PAGEON
0B97                   123          PAG
0B97                   124          .FI
0B97                   125  ;-----------------------------------------------------------
0B97                   126  AERR3:
0B97                   127  ; Pops sufficient address from after the FMLI call
0B97                   128  ;  to insure a correct return. Stores the a-reg into
0B97                   129  ;  asave, sets the carry and exits thru bfexit.
0B97                   130  ;-----------------------------------------------------------
0B97 20 F1 0A          131          JSR INCPP
0B9A 20 F1 0A          132          JSR INCPP
0B9D 20 F1 0A          133  AERR2   JSR INCPP
0BA0 20 F1 0A          134          JSR INCPP
0BA3 20 F1 0A          135  AERR1   JSR INCPP
0BA6 20 F1 0A          136          JSR INCPP
0BA9 85 F5             137  AERR0   STA ASAVE
0BAB 20 E8 0A          138          JSR BFSEC
0BAE 4C CB 0A          139          JMP BFEXIT
```

```
0BB1                ;
0BB1                ;-----------------------------------------------------
0BB1                MAPFMERR:
0BB1                ; Enter with a jump.
0BB1                ; Converts the filemanager error number to an FMLI
0BB1                ; number in a-reg. Sets the carry, and jmp to BFEXIT.
0BB1                ; Used only if there are no further argument
0BB1                ; addresses to pull after fmli call.
0BB1                ;-----------------------------------------------------
0BB1 AE C5 B5                LDX FMRETURN
0BB4 BD BA 0B                LDA MAPTABLE,X
0BB7 4C A9 0B                JMP AERRO
0BBA                ;
0BBA 00 00 0E       MAPTABLE .DA #$00,#$00,#EBADCALL
0BBD 0E 03 00                .DA #EBADCALL,#EPROTECT,#$00
0BC0 01 08 09                .DA #EOPEN,#EPLACE,#EIOERR
0BC3 03                      .DA #EPROTECT
0BC4                ;
0BC4                ;-----------------------------------------------------
0BC4                ZEROREC:
0BC4                ; Short routines to zero out areas of the filemanager
0BC4                ;  paramater list.
0BC4                ;-----------------------------------------------------
0BC4 A9 00                   LDA #$00
0BC6 8D BD B5                STA FMRECORD
0BC9 8D BE B5                STA FMRECORD+$01
0BCC 60                      RTS
0BCD                ZEROOFF:
0BCD A9 00                   LDA #$00
0BCF 8D BF B5                STA FMOFFSET
0BD2 8D C0 B5                STA FMOFFSET+$01
0BD5 60                      RTS
0BD6                ;
0BD6 16 00 16       ALLOCTBL .DA #!22,/!22,#!22,/!22
0BDA 34 00 61                .DA #!52,/!52,#!97,/!97
0BDE 61 01                   .DA #!353,/!353
0BE0                ;
0BE0                         .IF PAGEON
0BE0                         PAG
0BE0                         .FI
0BE0                ;-----------------------------------------------------
0BE0                FALLOC:
0BE0                ; Allocate a buffer. Called by almost everything,
0BE0                ; including catalog, delete, and rename.
0BE0                ; Any unused buffers, swapdown. Otherwise create a
0BE0                ; new buffer, adjust HIMEM down, and swapdown.
0BE0                ;-----------------------------------------------------
0BE0 AD 05 0A                LDA HEAPADR          ;find the heap
0BE3 85 FA                   STA AP
0BE5 AD 06 0A                LDA HEAPADR+$01
0BE8 85 FB                   STA AP+$01
0BEA A0 00                   LDY #$00             ;any buffers created?
0BEC B1 FA                   LDA (AP),Y
0BEE F0 12                   BEQ ALLOCNEW         ;no, skip to create one.
0BF0 48                      PHA
0BF1 20 AB 0C                JSR STATSAVE         ;save old file's status
0BF4 68                      PLA
0BF5 A8                      TAY
0BF6 B1 FA          ^1       LDA (AP),Y
0BF8 D0 03                   BNE *+5              ;yes, any available?
0BFA 4C 7C 0C                JMP ALLOCOLD         ; yes, skip to use it.
0BFD 88                      DEY
0BFE 88                      DEY
0BFF 88                      DEY
0C00 10 F4                   BPL <1
0C02 D0 20          ALLOCNEW BNE >0
0C04 AD C1 AA                LDA $AAC1            ;if first file ever
0C07 85 FC                   STA VP               ; copy s,d,v from RWTS
0C09 AD C2 AA                LDA $AAC2            ; as initial defaults.
0C0C 85 FD                   STA VP+$01
0C0E A0 01                   LDY #$01
```

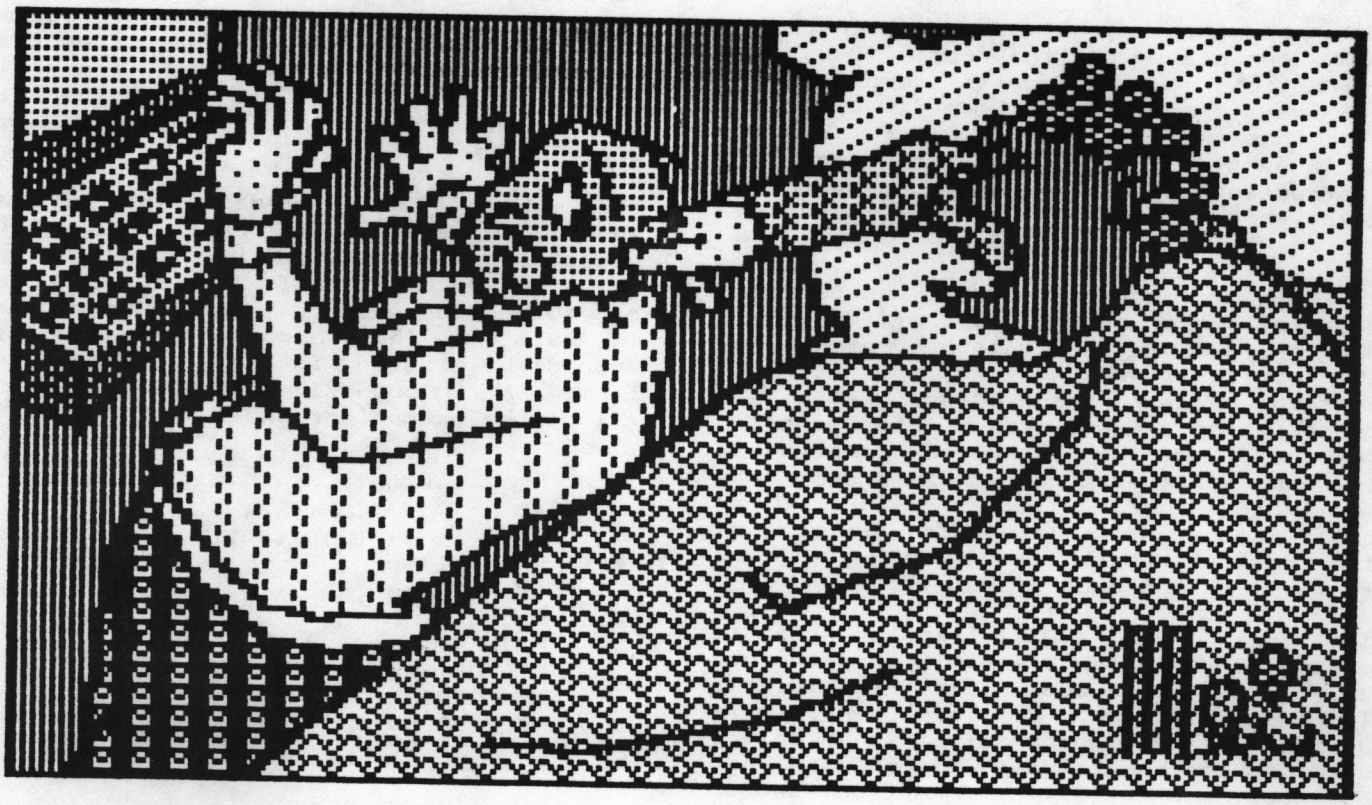
```
0C10 B1 FC      211           LDA (VP),Y
0C12 4A         212           LSR
0C13 4A         213           LSR
0C14 4A         214           LSR
0C15 4A         215           LSR
0C16 85 02      216           STA FCURSLOT
0C18 C8         217           INY
0C19 B1 FC      218           LDA (VP),Y
0C1B 85 01      219           STA FCURDRIV
0C1D C8         220           INY
0C1E B1 FC      221           LDA (VP),Y
0C20 49 FF      222           EOR #$FF
0C22 85 00      223           STA FCURVOL
0C24 A0 00      224 ^0        LDY #$00              ;create a new buffer.
0C26 B1 FA      225           LDA (AP),Y
0C28 D0 04      226           BNE >1
0C2A A9 01      227           LDA #$01              ;first buffer offset =1
0C2C D0 0B      228           BNE >2
0C2E 18         229 ^1        CLC                   ; else offset=offset+3
0C2F 69 03      230           ADC #$03
0C31 C9 17      231           CMP #!23
0C33 90 04      232           BLT >2                ;beyond top of heap
0C35 A9 0B      233           LDA #EBUFFERS         ; returns error.
0C37 38         234           SEC
0C38 60         235           RTS
0C39 91 FA      236 ^2        STA (AP),Y            ;update heap index
0C3B A8         237           TAY
0C3C 0A         238           ASL
0C3D 0A         239           ASL
0C3E 85 06      240           STA FCURFD            ;create new file desc.
0C40 C8         241           INY
0C41 B1 FA      242           LDA (AP),Y            ;update himem from heap
0C43 85 FE      243           STA HIMEM             ; and set new file status
0C45 85 EF      244           STA FSP               ; pointer.
0C47 C8         245           INY
0C48 B1 FA      246           LDA (AP),Y
0C4A 85 FF      247           STA HIMEM+$01
0C4C 85 F0      248           STA FSP+$01
0C4E 20 AB 0C   249           JSR STATSAVE          ;Copy defaults to buffer.
0C51 A0 04      250           LDY #$04
0C53 A2 00      251           LDX #$00
0C55 A5 EF      252 ^3        LDA FSP               ;use table to create
0C57 18         253           CLC                   ; correct addresses in
0C58 7D D6 0B   254           ADC ALLOCTBL,X        ; buffer.
0C5B 91 EF      255           STA (FSP),Y
0C5D E8         256           INX
0C5E C8         257           INY
0C5F A5 F0      258           LDA FSP+$01
0C61 7D D6 0B   259           ADC ALLOCTBL,X
0C64 91 EF      260           STA (FSP),Y
0C66 E8         261           INX
0C67 C8         262           INY
0C68 C0 0D      263           CPY #$0D
0C6A 90 E9      264           BLT <3
0C6C A0 06      265           LDY #$06              ;copy file descriptor
0C6E A5 06      266           LDA FCURFD            ; to buffer.
0C70 91 EF      267           STA (FSP),Y
0C72 C8         268           INY
0C73 A9 07      269           LDA #FRWSTAT          ;Restore  read/write
0C75 91 EF      270           STA (FSP),Y           ; flag in buffer.
0C77 A0 00      271           LDY #$00
0C79 B1 FA      272           LDA (AP),Y            ;recover index and fall
0C7B A8         273           TAY                   ; into:
0C7C            274 ALLOCOLD:
0C7C            275 ; We don't mark the buffer as in-use here, as in-use
0C7C            276 ;  only applies for FOPEN and FCREATE.
0C7C C8         277           INY
0C7D B1 FA      278           LDA (AP),Y            ;copy address of buffer
0C7F 85 EF      279           STA FSP               ; into FSP.
```

```
0C81 C8          280          INY
0C82 B1 FA       281          LDA (AP),Y
0C84 85 F0       282          STA FSP+$01          ;move buffer status to
0C86 20 A0 0C    283          JSR SWAPDOWN         ; zero page.
0C89 A0 1D       284          LDY #!29
0C8B A9 A0       285          LDA #" "             ;clear name
0C8D 91 04       286  ^1      STA (FCURNP),Y       ; to spaces.
0C8F 88          287          DEY
0C90 10 FB       288          BPL <1
0C92 18          289          CLC                  ;carry clear, no errors.
0C93 60          290          RTS
0C94             291  ;
0C94             292          .IF PAGEON
0C94             293          PAG
0C94             294          .FI
0C94             295  ;-------------------------------------------------------
0C94             296  SWAPUP:
0C94             297  ; Copy zero page status area up into file manager
0C94             298  ; paramater list.
0C94             299  ;-------------------------------------------------------
0C94 A0 0D       300          LDY #$0D
0C96 B9 00 00    301  ^1      LDA FCURVOL,Y
0C99 99 BF B5    302          STA FMVOLUME,Y
0C9C 88          303          DEY
0C9D 10 F7       304          BPL <1
0C9F 60          305          RTS
0CA0             306  ;
0CA0             307  ;-------------------------------------------------------
0CA0             308  SWAPDOWN:
0CA0             309  ; Copy the file status area from buffer to zero page.
0CA0             310  ;-------------------------------------------------------
0CA0 A0 15       311          LDY #$15
0CA2 B1 EF       312  ^1      LDA (FSP),Y
0CA4 99 00 00    313          STA FCURVOL,Y
0CA7 88          314          DEY
0CA8 10 F8       315          BPL <1
0CAA 60          316          RTS
0CAB             317  ;
0CAB             318  ;-------------------------------------------------------
0CAB             319  STATSAVE:
0CAB             320  ; Save the current values of the file's zero page
0CAB             321  ; status area into its buffer.
0CAB             322  ;-------------------------------------------------------
0CAB A0 15       323          LDY #$15
0CAD B9 00 00    324  ^0      LDA FCURVOL,Y
0CB0 91 EF       325          STA (FSP),Y
0CB2 88          326          DEY
0CB3 10 F8       327          BPL <0
0CB5 60          328          RTS
0CB6             329  ;
0CB6             330          .IF PAGEON
0CB6             331          PAG
0CB6             332          .FI
0CB6             333  ;-------------------------------------------------------
0CB6             334  PARSENAM:
0CB6             335  ; On entry, AP points to the filename to be parsed.
0CB6             336  ; FCURNP points to where the name proper is to be
0CB6             337  ; placed. Stripped vol,slot, and drive specifications
0CB6             338  ; are placed into proper zero page spots. No test
0CB6             339  ; for proper values is made, as various Hard Disks
0CB6             340  ; rely on abnormal specs for drive and slot. A
0CB6             341  ; prescan should be preformed by calling routine, if
0CB6             342  ; length, at, etc. specifications are expected.
0CB6             343  ;-------------------------------------------------------
0CB6 A0 00       344          LDY #$00
0CB8 20 FC 0C    345  ^2      JSR PARS1            ;transfer name
0CBB F0 12       346          BEQ >5               ; up to comma,
0CBD 91 04       347          STA (FCURNP),Y       ; cr, or 00.
0CBF C8          348          INY
0CC0 C0 1E       349          CPY #!30             ;30 chars max.
0CC2 90 F4       350          BLT <2
```

```
0CC4 C8          351  PARS0    INY                ;Over 30 truncates
0CC5 F0 07       352           BEQ >4             ; and scans for
0CC7 20 FC 0C    353  PARS00   JSR PARS1          ; comma or EOS.
0CCA F0 03       354           BEQ >5
0CCC D0 F6       355           BNE PARS0
0CCE 60          356  ^4       RTS
0CCF C8          357  ^5       INY                ;Comma: fetch next
0CD0 F0 FC       358           BEQ <4             ; character.
0CD2 20 FC 0C    359           JSR PARS1
0CD5 20 30 0D    360           JSR TOUPPER        ;Normalize to uppercase.
0CD8 C9 D6       361           CMP #"V"
0CDA D0 08       362           BNE >6             ;Volume:
0CDC 20 0C 0D    363           JSR PARSNUM        ; Overflow ignored.
0CDF 85 00       364           STA FCURVOL        ; ie, 256=0, 257=1...
0CE1 4C C7 0C    365           JMP PARS00
0CE4 C9 D3       366  ^6       CMP #"S"           ;Slot:
0CE6 D0 08       367           BNE >7
0CE8 20 0C 0D    368           JSR PARSNUM
0CEB 85 02       369           STA FCURSLOT
0CED 4C C7 0C    370           JMP PARS00
0CF0 C9 C4       371  ^7       CMP #"D"           ;Drive:
0CF2 D0 D0       372           BNE PARS0
0CF4 20 0C 0D    373           JSR PARSNUM
0CF7 85 01       374           STA FCURDRIV
0CF9 4C C7 0C    375           JMP PARS00         ;Other letters ignored.
0CFC             376  ;                           Note: a space after
0CFC             377  ;                             the comma, skips to
0CFC             378  ;                             the next comma.
0CFC             379  ;
0CFC B1 FA       380  PARS1    LDA (AP),Y         ;Subroutine to fetch
0CFE F0 09       381           BEQ >1             ; next char.
0D00 09 80       382           ORA #$80           ;Normalize hi-bit on.
0D02 C9 8D       383           CMP #CR            ;Cr or 00 abort.
0D04 F0 03       384           BEQ >1
0D06 C9 AC       385           CMP #","           ;Compare to comma.
0D08 60          386           RTS
0D09 68          387  ^1       PLA                ;Abort, pull top rts.
0D0A 68          388           PLA                ; and return.
0D0B 60          389           RTS
0D0C             390  ;
0D0C A2 00       391  PARSNUM   LDX #$00           ;Dec to 1 byte Hex.
0D0E C8          392  ^0       INY                ; Hex in x-reg.
0D0F F0 F8       393           BEQ <1
0D11 B1 FA       394           LDA (AP),Y         ;Get char and
0D13 29 7F       395           AND #$7F           ; strip hi-bit.
0D15 38          396           SEC                ;Subtract ascii zero.
0D16 E9 30       397           SBC #'0'
0D18 90 14       398           BCC >2             ; underflow non-digit.
0D1A C9 0A       399           CMP #$0A           ; >9 non digit.
0D1C B0 10       400           BCS >2
0D1E 85 FC       401           STA VP             ;save digit.
0D20 8A          402           TXA                ; Hex=Hex*10
0D21 0A          403           ASL
0D22 85 FD       404           STA VP+$01
0D24 0A          405           ASL
0D25 0A          406           ASL
0D26 18          407           CLC
0D27 65 FD       408           ADC VP+$01
0D29 65 FC       409           ADC VP             ; Hex=Hex+Digit.
0D2B AA          410           TAX
0D2C 90 E0       411           BCC <0             ;no overflow, continue.
0D2E 8A          412  ^2       TXA                ;Exit, move Hex to a-reg.
0D2F 60          413           RTS
0D30             414  ;
0D30             415  ;-------------------------------------------------------------
0D30             416  TOUPPER:
0D30             417  ; Convert a-reg to upper case. Expects hi bit set on
0D30             418  ; entry. x,y preserved.
0D30             419  ;-------------------------------------------------------------
0D30 C9 E1       420           CMP #"a"
0D32 90 07       421           BLT >1
```

```
OD34 C9 FB       422              CMP #"z"+$01
OD36 B0 03       423              BGE >1
OD38 38          424              SEC
OD39 E9 20       425              SBC #$20
OD3B 60          426  ^1          RTS
OD3C             427  ;
OD3C             428              .IF PAGEON
OD3C             429              PAG
OD3C             430              .FI
OD3C             431  ;-----------------------------------------------------------------
OD3C             432  BFGETC:
OD3C             433  ; Calling format: JSR CALLFMLI
OD3C             434  ;                  BYT FGETC
OD3C             435  ;                  ADR FILEDESCRIPTOR
OD3C             436  ; Fetches next byte from opened file. Returns char
OD3C             437  ;  in a-reg. Carry set on error. A-reg = 00 and carry
OD3C             438  ;  set if EOF.
OD3C             439  ;-----------------------------------------------------------------
OD3C 20 75 OD    440              JSR GETFDO
OD3F A5 07       441              LDA FRWSTAT          ;check read/write status
OD41 C9 01       442              CMP #$01             ; error if write only.
OD43 D0 05       443              BNE >1
OD45 A9 0C       444              LDA #EWRITE
OD47 4C A9 0B    445              JMP AERRO
OD4A A9 03       446  ^1          LDA #$03             ;opcode=3 for read/write
OD4C 8D BB B5    447              STA FMOPCODE
OD4F A9 01       448              LDA #$01             ;subcode=1 read 1 byte
OD51 8D BC B5    449              STA FMSUBCOD
OD54 20 FD AA    450              JSR CALLFM           ;call the file manager
OD57 AD C3 B5    451              LDA FMBYTE           ;Return the byte whether
OD5A 85 F5       452              STA ASAVE            ; or not an error.
OD5C 90 03       453              BCC >2
OD5E 4C B1 0B    454              JMP MAPFMERR         ;error exits
OD61 4C DD 0D    455  ^2          JMP COMCLCEX
OD64             456  ;
OD64             457              .IF PAGEON
OD64             458              PAG
OD64             459              .FI
OD64             460  ;-----------------------------------------------------------------
OD64             461  GETFD2:
OD64             462  ; Vectors to calling FDTOFMGR, which exit on error
OD64             463  ; thru the correct AERR routine.
OD64             464  ;-----------------------------------------------------------------
OD64 20 7D OD    465              JSR FDTOFMGR
OD67 90 03       466              BCC >0
OD69 4C 9D 0B    467              JMP AERR2
OD6C 60          468  ^0          RTS
OD6D 20 7D OD    469  GETFD1      JSR FDTOFMGR
OD70 90 FA       470              BCC <0
OD72 4C A3 0B    471              JMP AERR1
OD75 20 7D OD    472  GETFDO      JSR FDTOFMGR
OD78 90 F2       473              BCC <0
OD7A 4C A9 0B    474              JMP AERRO
OD7D             475  ;
OD7D             476              .IF PAGEON
OD7D             477              PAG
OD7D             478              .FI
OD7D             479  ;-----------------------------------------------------------------
OD7D             480  FDTOFMGR:
OD7D             481  ; Fetch the file descriptor & normalize.  Copys the
OD7D             482  ;  status address in heap to FSP. Returns with the
OD7D             483  ;  heap's address in AP, and files offset in a-reg.
OD7D             484  ; Initializes file manager paramater list.
OD7D             485  ;  Illegal file descriptors abort to bfexit.
OD7D             486  ;-----------------------------------------------------------------
OD7D 20 F8 0A    487              JSR PULLADR          ;Is the file descriptor
OD80 C5 06       488  FDTO2       CMP FCURFD           ; current file?
OD82 F0 05       489              BEQ >1
OD84 48          490              PHA                  ;No, save the current
OD85 20 AB 0C    491              JSR STATSAVE         ; file's status.
OD88 68          492              PLA
OD89 4A          493  ^1          LSR                  ;normalize to 0 base.
OD8A 4A          494              LSR
OD8B F0 13       495              BEQ >2
```

```
OD8D 48          496           PHA                    ;save for return.
OD8E AC 05 0A    497           LDY HEAPADR
OD91 84 FA       498           STY AP                 ;copy heapadr to ap.
OD93 AC 06 0A    499           LDY HEAPADR+$01
OD96 84 FB       500           STY AP+$01
OD98 A0 00       501           LDY #$00
OD9A D1 FA       502           CMP (AP),Y             ;If file descriptor
OD9C F0 06       503           BEQ >1                 ; greater than allocated
OD9E 90 04       504           BLT >1                 ; file buffers
ODA0 A9 0E       505  ^2        LDA #EBADCALL          ;illegal descriptor is
ODA2 38          506           SEC                    ; "bad system call".
ODA3 60          507           RTS
ODA4 A8          508  ^1        TAY                    ;Also check against table
ODA5 B9 BD 0D    509           LDA OKFDTBL,Y          ; of possible values, and
ODA8 F0 F6       510           BEQ <2                 ; exit error if bad.
ODAA C8          511           INY                    ;Copy file pointer to
ODAB B1 FA       512           LDA (AP),Y             ; fsp.
ODAD 85 EF       513           STA FSP
ODAF C8          514           INY
ODB0 B1 FA       515           LDA (AP),Y
ODB2 85 F0       516           STA FSP+$01
ODB4 20 A0 0C    517           JSR SWAPDOWN           ;initializer file mgr.
ODB7 20 94 0C    518           JSR SWAPUP             ; paramater list.
ODBA 68          519           PLA                    ;restore offset and
ODBB 18          520           CLC
ODBC 60          521           RTS                    ; return.
ODBD             522  ;
ODBD             523  ; I decided to use a table, since the legal normalized
ODBD             524  ;  file descriptor values are 1,4,7,10,13,16,19,23.
ODBD             525  ;
ODBD 00 01 00    526  OKFDTBL   HEX 0001000001000001
ODC5 00 00 01    527           HEX 0000010000010000
ODCD 01 00 00    528           HEX 0100000100000100
ODD5             529  ;
ODD5             530  ; to be continued ...
```



Use VisiCalc to Compute your Federal Tax

-Don Scellato

For the past few years, I've used VisiCalc to compute my Federal and State Income Tax Returns. Each year I've learned a few more tricks with VisiCalc to make it easier to perform this annual task that most of us approach with some dread. VisiCalc like all other electronic spreadsheet programs takes some of the sting out of tax preparation by doing some of the routine and non-routine math required by the various tax forms and schedules. The methods discussed in this article for handling the math and setting up the spreadsheet could just as easily be used on Magicalc, Multiplan, Supercalc, or whatever.

The example that appears in this article was developed by Stan Dinsmore and I at meetings of the SFAC VisiCalc Interest Group. This particular template is included in VisiCalc Template Disk No.2 which is currently being sold by the San Francisco Apple Core. The template has been written purposely to fit the 40 column screen except for one section which stretched out to about 90 columns. The reason for this particular column fit was to display the spreadsheet and make entries to the spreadsheet without requiring a lot of scrolling right and left by the user.

This template may not be turned in to the IRS in place of the Form 1040. The infomation contained on the template must be transferred to a Form 1040. If the user has any doubt as to what is required on a particular line on the tax form, the instruction booklet which came with the taxpayer's tax forms should be consulted.

If this document is unclear, the taxpayer may call the nearest IRS office or may obtain further information by picking up the publications issued by the IRS which pertain to such particular questions as Depreciation, Exemptions, Investment Tax Credit, Income, and so on. Publication 17 is one of the more complete documents available from the IRS. Other publications are listed in the back of the booklet which came with your tax forms or in the back of Pub 17.

The Form 1040 (1983) template which is presented with this article is described as the template appears on SFAC VisiCalc Template Disk No. 2. This template is divided into a number of parts which are used to gather taxpayer information, set up wage and salary information, set up tax withholding information, compute total income, compute total adjustments to income, carry information to Schedule A which is included within the body of the tax form template, compute itemized (Schedule A) deductions, compute taxable income, compute total taxes due, compute tax credits, compute other taxes due, compute total payments made by the taxpayer, and finally to compute overpayment or amount of tax owed.

Taxpayer information which is required on Form 1040 should be entered in columns B,C,and D of rows 23 through 32. If the cell width is exceeded, the user should move to the adjoining cell by using the right arrow key with the cursor direction set to the horizontal mode. This portion is included in case the user of the template prepares taxes other than his own. Filing status is entered by placing a "1" in C37, C38, C39, C40, or C42 as required by the user. Cell C41 is used to enter the name of the child or children which qualify a taxpayer as a Head of Household.

Cell C43 is used to enter the name of the deceased spouse which qualifies a taxpayer as a Qualifying Widow(er). The information which is gathered in this section of the template is used in the Schedule A portion of the template to compute the reduction to deductions required by marital status. Exemption information is gathered in lines 48 through 72 by entering the number "1" where appropriate in cells C48, C49, C50, C51, C52, and C53.

The actual number of dependent children should be entered in celll C56. Their names shoud be listed in columns B and C of lines 57 through 61. The actual number of "Other Dependents" should be entered in cell C65. Their names and relationship should be listed in cloumns B and C of lines 66 through 71. Cell C72 is used to all of the exemptions shown.

```
        A          B          C
    ----------------------------------
21              TAXPAYER DATA
22        ENTER TAXPAYER DATA BELOW
23  TAXPYER NAME
24  ADDRESS::::>
25  CITY:::::::>
26  STATE::::::>
27  ZIP::::::::>
28  SOC SEC NO:>
29  OCCUPATION:>
30  SPOUSE NAME>
31  SOC SEC NO:>
32  OCCUPATION:>
33  ----------------------------------
```

```
34              FILING STATUS
35          ENTER FILING STATUS BELOW
36        PLACE A "1" IN PROPER BOX
37   SINGLE::::::::::::::::::>          0
38   MARRIED FILING JOINT:::>           0
39   MARRIED FILING SEPARATE>           0
40   HEAD OF HOUSEHOLD::::::>           0
41   CHILD'S NAME::::::::::::>
42   QUALIFYING WIDOW(ER):::>
43   YEAR SPOUSE DIED:::::::>
44   ----------------------------------
45              EXEMPTIONS
46          ENTER EXEMPTIONS BELOW
47        PLACE A "1" IN PROPER BOX
48   YOURSELF:::::::::::::::::>         0
49   65 OR OVER:::::::::::::::>         0
50   BLIND:::::::::::::::::::>
51   SPOUSE:::::::::::::::::::>         0
52   65 OR OVER:::::::::::::::>
53   BLIND:::::::::::::::::::>
54   ENTER NUMBER OF DEPENDENT CHILDREN &
55   LIST NAMES IN LINES PROVIDED::::::::
56   NUMBER OF CHILDREN:::::>           0
57   LIST NAMES:>.....
58              .....
59              .....
60              .....
61              .....
62   ENTER NUMBER OF OTHER DEPENDENTS &
63   LIST NAMES, RELATIONSHIP, & MONTHS
64   IN HOUSEHOLD IN LINES PROVIDED::::::
65   NUMBER OTHER DEPENDENTS>           0
66   NAME:::::::>.....
67   RELATIONSHIP.....
68   NAME:::::::>.....
69   RELATIONSHIP.....
70   NAME:::::::>.....
71   RELATIONSHIP.....
72   TOTAL EXEMPTIONS::::::::           0
```

Lines 73 through 90 of the template are used to gather information regarding gross wages paid to the taxpayer, Federal Tax withheld, FICA withheld from the taxpayer's wages and the spouse's wages separately, state tax withheld, and SDI (State Disability Insurance) withheld. The wages shown on the W2 for both parties should be entered in column C between lines 78 and 86. The formula in C88 sums the entries in cell C78 through C86 (@SUM(C78...C86).

Federal Tax withheld from the wages of both parties shouldbe entered in column D from line 78 through line 86. The formula in cell D88 sums the amounts shown in the range from D78 to D86 (@SUM(D78...D86). The taxpayer's FICA withholding amounts should be entered in column E from line 78 through line 86. The sum of these amounts is derived in E88 (@SUM(E78...E86).

If this amount exceeds the maximum FICA liability of the taxpayer, the excess is computed in cell E90. The spouse's FICA withholding amounts should be entered in column F in the same fashion that this information was entered for the taxpayer in column E. The same calculation of excess withholding is computed for the spouse in F90 as was done for the taxpayer in E90. State withholding amounts should be entered in column G from line 78 through line 86. The formula in G88 sums these amounts (@SUM(G78...G86). Once all of this wage and withholding information has been entered, it is transferred to the line on the tax form or on the Schedule A where it is needed.

```
     A          B          C          D          E          F          G
73   ----------------------------------
74   WAGE, SALARY, & WITHHOLDING DATA
75                                                  TAXPAYER   SPOUSE
76   TAXPAYER   EMPLOYER   GROSS WAGES  FED W'HELD  FICA W'HLD FICA W'HLD STATE W'HLD
77   --------   --------   -----------  ----------  ---------- ---------- ----------
78                             0.00        0.00        0.00                  0.00
79                             0.00        0.00        0.00                  0.00
80
81
82
83
84
85
86
87                        ------------------------------------------------------------
88                             0.00        0.00        0.00       0.00       0.00
89   FICA LIMIT ::::::::::::::::::::::::::::::::::::::::::>       2391.90    2391.90
90   EXCESS FICA ::::::::::::::::::::::::::::::::::::::::::>          0.00       0.00
91   ------------------------------------------------------------------------------
```

Lines 92 through 111 of the template are used to compute Total Income. Cell C93 brings down the total wages shown in C88 (+C88). The user enters total interest manually into C94. He also enters total taxable dividends manually into C95. The dividend exclusion is entered into C96. This amount should be $200 for a single taxpayer and $400 for married taxpayers filing jointly. The exclusion should be no greater than the total dividends if the dividends are less than $200 and $400 respectively.

The remainder of income information should be entered on the appropriate lines in this section. The formula in C112 totals all of the income (@SUM(C93...C94,C97...C104,C106...C108,C110...C111). Adjustments to income must be entered in cells C115 through C123 as appropriate. The formula in cell C124 totals all of the adjustments (@SUM(C115...C117,C119...C123).

```
      A          B          C
 91  ------------------------------------
 92              FORM 1040A
 93    7 - WAGES,SAL, & TIPS>        0.00
 94    8 - INTEREST INCOME::>        0.00
 95   9A - DIVIDEND INCOME::>        0.00
 96   9B - EXCLUSION::::::::>        0.00
 97   9C - TAXABLE DIVIDENDS>        0.00
 98   10 - STATE TAX REFUNDS>        0.00
 99   11 - ALIMONY REC'D::::>
100   12 - SCH C BUS INC/LSS>
101   13 - SCH D CAPITAL GNS>
102   14 - 40% CAP GAINS DST>
103   15 - FORM 4797 GNS/LSS>
104   16 - TAXABLE PENS/ANN:>
105  17A - OTHER PENS/ANN:::>        0.00
106  17B - TAXABLE AMOUNT:::>
107   18 - SCH E RNTS/ROY:::>
108   19 - SCH F FARM INC:::>
109  20A - TOT UNEMP INC REC>        0.00
110  20B - TAXABLE AMOUNT:::>
111   21 - OTHER INCOME:::::>        0.00
112   22 - TOTAL INCOME:::::>        0.00
113  ------------------------------------
114  ADJUSTMENTS TO INCOME
115   23 - FORM 3903 MV EXP:>
116   24 - FORM 2106 EMP BUS>        0.00
117  25A - PAYMENTS TO IRA::>        0.00
118  25B - 1984 PYMNT IN 25A>
119   26 - PAYMENTS TO KEOGH>
120   27 - PEN FOR EARLY W/D>
121   28 - ALIMONY PAID:::::>
122   29 - MARRIED CPL DED::>
123   30 - FORM 2440 DIS EXC>
124   31 - TOTAL ADJUSTMENTS>        0.00
125  ------------------------------------
126   32 - ADJUSTED GROSS....
127              INCOME:::::>        0.00
128  ====================================
```

Lines 128 through 196 of the template are devoted to Schedule A. The appropriate amounts from the taxpayer's records should be entered in the appropriate line of the form. Only those cells described below as containing formulas should not be entered by the user. Cell C131 computes 1% of Adjusted Gross Income which is computed in cell C127 (+C127*.01). Cell C131 computes 5% of the taxpayer's Adjusted Gross Income (+c127*.05). Cell C144 is used to compute total allowable medical deduction by using the IF function and one of the logical operators (@IF(C143>C142,0,+C142-C143).

Thus, if the amount in C143 (5% of Adjusted Gross Income) is greater than the total medical deductions shown in C142, a zero is inserted in C144. Otherwise, the amount in C143 is subtracted from the amount in C142 to arrive at the proper deduction amount.

Cell C146 brings down the total state income tax withheld from cell G88 (+G88). Cell C151 does the same thing with total SDI withheld (+H88). The formula in C154 sums the amounts listed in the Tax Section (@SUM(C146..C153). The formula in C166 sums all of the interest amounts shown in cells C156 through C165 (@SUM(C156...C165).

The formula in C175 does the same thing for contributions (@SUM(C168...C174). The formula in C186 does the same thing for miscellaneous deductions (@SUM(C180...C185). Cell C189 totals all of the deductions from all categories (+C144+C154+C166+C175+C177+C186). The formula in C192 uses the entries from the filing status area discussed earlier to determine whether $2300 should be subtracted from the total deductions or $0 should be deducted as this status is not the valid status for the taxpayer.

The IF statement is used along with the OR statement to define the proper response (@IF(@OR(C37=1,C40=1),-2300,0). This sets up the reduction for single taxpayers or Head of Household. The formula in C193 sets up the same logic for Marred Filing Joint or Qualifying Widow(er) (@IF(@OR(C38=1,C42=1),-3400,0). The fomula in C194 sets up the same logic for Married Filing Separate (@IF(C39=1,-1700,0). Cell C196 totals out the deductions from Schedule A (@IF(C189+@SUM(C192...C194)>=0,C189+@SUM(C191...C194),0). This total is hen transferred to the proper place on the 1040.

```
      A          B          C
128  ====================================
129  SCHEDULE A COMPUTATION
130    1 - MEDICINE & DRUGS:>        0.00
131    2 - 1% LINE 32(33):::>        0.00
132    3 - LINE 2 - LINE 1::>        0.00
133    4 - OTHER MED & DENT
134      4A - DOCTORS, DENT,
135      NURSES, HOSPITALS::>        0.00
136      4B - TRANSPORTATION>        0.00
137      4C - OTHER - LIST
138      ....................        0.00
139      ....................        0.00
140      ....................
141      ....................
142   5 - ADD LINES 3 - 4C::>        0.00
143   6 - 5% LINE 32(33)::::>        0.00
144   7 - LINE 5 - LINE 6:::>        0.00
145  ------------------------------------
146   8 - ST & LOC INC TAX::>        0.00
147   9 - REAL ESTATE TAX:::>        0.00
148  10A - GENERAL SALES::::>        0.00
149  10B - GEN SLS - M/VEH::>
150  11 - OTHER LIST -------
151       S.D.I..............        0.00
152       ...................
153       ...................
154  12 - ADD LINES 8 - 11::>        0.00
155  ------------------------------------
```

```
156  13A - HOME MORT INT -INS        0.00
157    B - HOME MORT INT -IND
158    IF IND SHOW NAME/ADD--
159    .....
160    .....
161  14 - CRED CDS & CHG ACC)        0.00
162  15 - OTHER - LIST ------
163     .....
164     .....
165     .....
166  16 - ADD LINES 13A - 15)        0.00
167  ---------------------------------
168  17A - CSH CONTRIBUTIONS)        0.00
169    B - CSH CONT OF $3000+
170        TO ONE ORG - LIST-
171        .....
172        .....
173  18 - OTHER THAN CASH:::)
174  19 - CARRYOVER PRIOR YR)
175  20 - ADD LINES 17 - 19:)        0.00
176  ---------------------------------
177  21 - TOTAL CAS & THEFT:)
178       ATTACH FORM 4684...
179  ---------------------------------
180  22 - UNION & PROF DUES:)
181  23 - TAX RET PREP FEE::)        0.00
182  24 - OTHER LIST -------
183       .....
184       .....
185       .....
186  25 - ADD LINES 22 - 24:)        0.00
187  ---------------------------------
188  26 - ADD 7,12,16,20,21,
189       & 25:::::::::::::::)        0.00
190
191  27 - DEDUCT FOR STATUS:)
192       SINGLE/HD OF HHLD:)        0.00
193       FILE JNT/WIDOW(ER))        0.00
194       MARRIED FLG SEP:::)        0.00
195
196  28 - LINE 26 - LINE 27:)        0.00
197  =================================
```

The final portion of the template is used to arrive at Taxable Income, enter Tax Due, figure credits, enter Other Taxes, enter payments, compute refund or compute Amount Due. Line 200 brings down the amount shown as Adjusted Gross Income on line 127 (+C127). Cell 201 brings down the total Schedule A Deductions (+C196).

If no Itemized Deductions are taken, the taxpayer may elect to take the allowed amount for contributions in C202. The formula in C203 takes the amount in C201 or the amount in C202 (@IF((C201=0),(+C200-C202),(+C200-C201))). The formula in C204 multiplies the total exemptions shown in C72 by $1000 (+C72*1000). The formula in C205 derives Taxable Income (+C203+C204). The user should enter the tax from the Tax Table in cell C208. Additional taxes should be entered in C211. The formula in C212 computes Total Taxes (+C208+C211). Credits must be entered as required in lines 215 through 221. The formula in C222 computes total credits (@SUM(C215...C221)).

The Balance of taxes to credits is computed in C224 (+C212-C222). Other taxes are entered in lines 227 through 232. The sum of these taxes is computed in C234 and is added to the Balance computed earlier (+C222+(@SUM(C227...C232)). Total Income Tax Withheld is brought down from D88 and entered into C237 (+D88). Excess FICA is brought down to C241 from cells E90 and F90 (+E90+F90). Total payments are computed in C245 (@SUM(C237...C243). The amount overpaid is calculated in C247 (@IF(C245>C234,C245-C234,0)).

The user must insert the amount to be refunded in C248. Cell C249 then uses this amount to aoutmatically calculate the amount to be applied to 1984 taxes (@IF(C248>0,C247-C248,0)). C250 is used to compute tax due (@IF(C234>C245,C234-C245,0)).

```
          A          B          C
197  ==================================
198  TAX COMPUTATION
199
200   33 - AMOUNT EX LINE 32)        0.00
201  34A - SCH A DEDUCTION::)        0.00
202  34B - CONT IF NO IT DED)        0.00
203   35 - LINE 33-34A OR 34B        0.00
204   36 - LINE 6E X 1000:::)        0.00
205   37 - TAXABLE INCOME:::)        0.00
206   38 - TAX EX TABLE....-
207        EX X,Y,Z...OR SCH
208        G.....:::::::::::)        0.00
209   39 - ADDITIONAL TAXES--
210        EX 4970...:4972...
211        5542...:SEC 72...)
212   40 - TOTAL TAXES::::::)        0.00
213  ----------------------------------
214  CREDITS
215   41 - CRED FOR ELD-R&RP)
216   42 - FRGN TX CRED-1116)
217   43 - INVST CRED -3468:)
218   44 - PART CRD - POLIT:)
219   45 - CHILD CARE:::::::)
220   46 - JOBS CRED - 5884:)
221   47 - RES ENERGY -5695:)
222   48 - TOTAL CREDITS::::)        0.00
223  ----------------------------------
224   49 - BALANCE::::::::::)        0.00
225  ----------------------------------
226  OTHER TAXES
227   50 - SLF EMP TX-SCH SE)
228   51 - ALT MIN TX - 6251)        0.00
229   52 - TAX FM RECAP-4255)
230   53 - FICA ON TIPS-4137)
231   54 - UNC EMP FICA/RRTA)
232   55 - TAX ON IRA - 5329)
233  ----------------------------------
234   56 - TOTAL TAX::::::::)        0.00
235  ----------------------------------
```

236	PAYMENTS	
237	57 - FED INC TAX W/H::)	0.00
238	58 - 83 EST TX PYMNTS:)	
239	59 - EARNED INC CRED::)	
240	60 - AMOUNT PD W/4868:)	
241	61 - EXCESS FICA/RRTA:)	0.00
242	62 - CRED TX ON FUEL::)	
243	63 - R.I.C. CRD -2439:)	
244	----------	
245	64 - TOTAL PAYMENTS:::)	0.00
246	----------	
247	65 - AMOUNT OVERPAID::)	0.00
248	66 - AMOUNT TO REFUND:)	0.00
249	67 - AMT TO APP TO 84:)	0.00
250	68 - AMOUNT OWED::::::)	0.00
251	----------	

The template that I've described may be used for tax preparation. If any user finds an incorrect formula or can think of another way to accomplish some of the calculations contained on this template, let me know care of the Editor of the Cider Press or come to a VisiCalc Interest Group meeting. The formulas have been shown to allow the reader to set up his own template if he doesn't wish to purchase the Template Disk. I'll try to present a template dealing with tax table computations in the next issue.

VisiCalc Interest Groups

-Don Scellato

The San Francisco VisiCalc Interest Group met at 7:30 P.M. on January 10, 1984, at the Clubhouse. Those in attendance discussed the 1983 Income Tax Template which is included on SFAC VISICALC TEMPLATE DISK NO. 2 which is now available at the Club's Sales Table for $10. A template covering ACRS depreciation was also demonstrated.

A VisiCalc DIF file sort program was demonstrated. The next meeting of the San Francisco VisiCalc Interest Group was held at 7:30 P.M. at the Clubhouse on Thursday, February 16. The attendees discussed tax applications further. Larry Nebron shared a template which combined 1983 Federal computations with 1983 California computations.

The debugged Tax Table for the Federal Return was presented and discussed. Both of these templates and another which incorporates the Tax Table with the Form 1040 will be released on SFAC VisiCalc Template Disk No. 3 at the March meeting. The next meeting of the San Francisco group will be held at 7:30 P.M. at the Clubhouse on Tuesday, March 13.

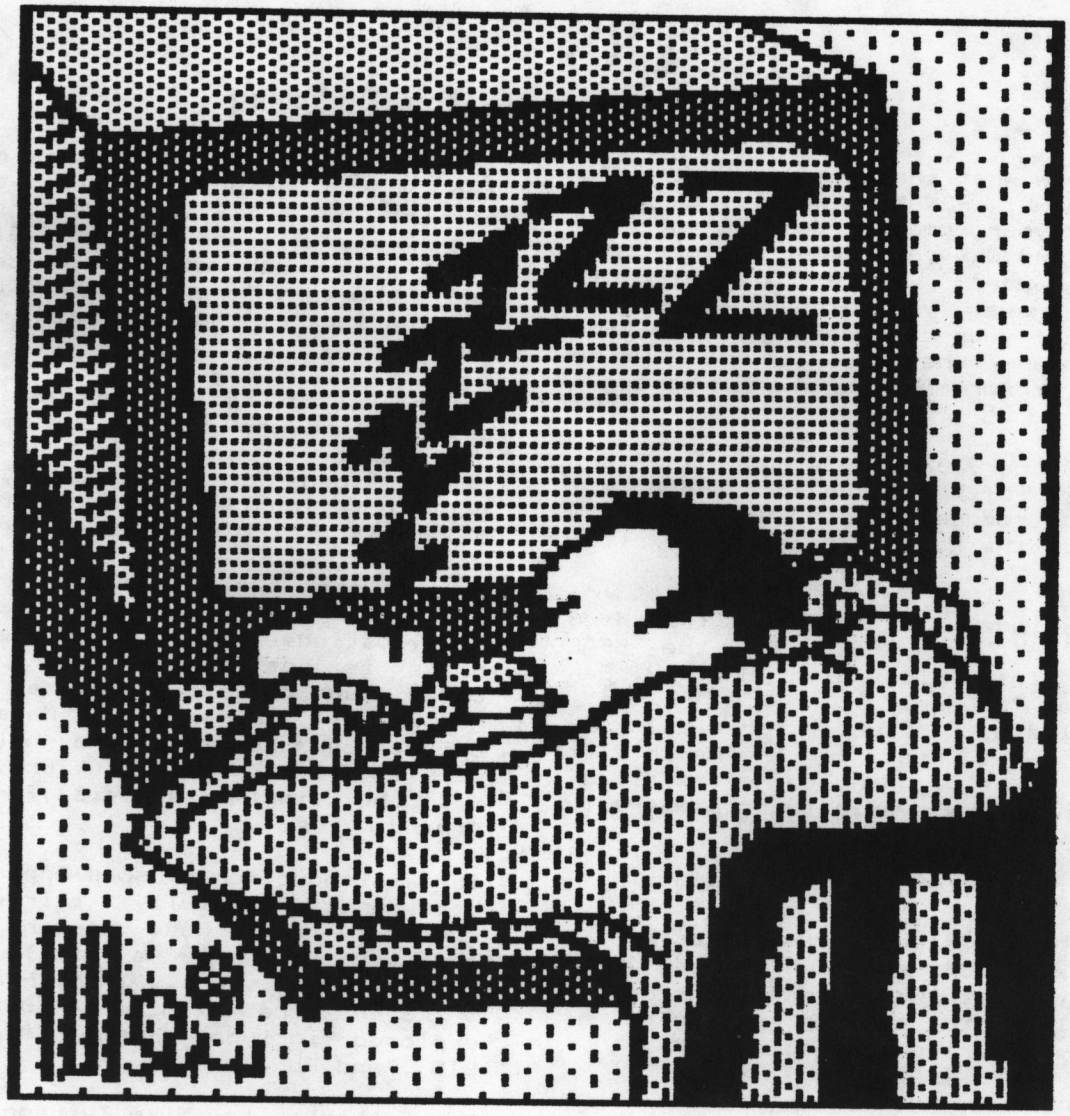
The East Bay VisiCalc Interest Group met at Breuner's in Pleasant Hill at 7:00 P.M. on January 26. Two applications dealing with using VisiCalc to compute income tax based on tax tables were discussed. One application used the Boolean Logic functions to select the proper tax for a particular filing status. The second method employed the CHOOSE command.

All of the commands were demonstrated for the first time users who attended the meeting. The next East Bay meeting was held at Breuner's on the second floor on Monday, February 27, at 7:00 P.M. Stan Dinsmore presented a template which computes California State Sales Tax for a small business. The combined Tax Table and Form 1040 was demonstrated and debugged at this meeting.

After some minor revisions suggested by those in attendance, it was tested and found to compute taxes correctly. When the results were compared with the Federal Tax Table, only a few errors were found. In those few cases where errors were detected, the results were within $1 of the tax table. The next East Bay Group meeting will be held at 7:00 P.M. on Monday, March 26, at Breuner's Furniture Store in the second floor meeting room.

The users of any spreadsheet are invited to attend these meetings. We can help you with the commands and applications of almost any electronic spreadsheet. Beginners are also invited. We'll take the time to explain and demonstrate cammands and features of VisiCalc and other spreadsheets.

If you've written any application and would like to share it with others, bring it along. If you've gotten part way through an application and are stuck, come to the meeting. We can help to solve the problem that's hanging you up. Be sure to bring a few blank disks because we usually have a few applications or utility programs to share.



INTEREST GROUP LEADERS

Marin CP/M	Jim Ayers	383-0473(day) 383-5759(eves)
S.F. CP/M	Ron Dell'Aquila	566-8460
East Bay	Molly Mitchell	526-9541
Dot-Matrix Printers	Michael Fischer	557-1250(day) 933-4435(eves)
Forth	Dudley Ackerman	626-6295
Marin Graphics	Reed Harris	924-7221
	Bob Sachs	332-3331
Nu'ers	Kathy Rudovsky	756-6480
Pascal	Mike Schnapp	386-2238
Product Review	Ben Lai	564-7773
VisiCalc et al	Don Scellato	672-2051
S.F.A.C. Clubhouse	* * *	566-2342

NUERS

Do you know what the commands NEW, HOME, INIT, CATALOG, RENAME, DELETE, LOAD and SAVE make your computer do? Do you know how, when and why you format a diskette? Have you ever written a mini-BASIC program? Have you ever changed the Volume # of your diskettes? Do you know what it means when you get a "write protected" error message on your screen?

Michael Fischer, January's guest speaker, covered all of these subjects and more! He also introduced Apple Computer's new "PRO DOS", the Koala Pad and AppleWriter II.

NUER'S meetings are held on the third Wednesday of each month at the SFAC Clubhouse, 1515 Sloat Blvd. Suite 2, San Francisco at 7:30 pm.
Don Scellato, Club Librarian, is scheduled guest host for the March 21st meeting. He is planning to review all of the many programs available to members in the SFAC Library.

URGENT!! HELP WANTED!! URGENT!!

Because there is so very much for a NUER (or even an OLDER) to learn and the Apple provides such varied fields of interest, I would like to set up a series of mini-seminars during the month of May.

Topics to be covered would include, Intro to the Apple Computer, Intro to Graphics on the Apple, Intro to Assembly Language, Intro to Applesoft BASIC, Intro to Pascual, Intro to Forth, Intro to Logo, Business Applications for the Apple and more.

If you have any knowledge or interests in sharing information on any of the forementioned topics or others, please contact Kathy Rudovsky as soon as possible. Please don't assume somebody else will call - YOU CALL!

If you are interested in attending any of these seminars sign-up sheets will be posted at the General Meetings or contact Kathy Rudovsky.

...from pg. 5

to insert subroutines into a program from a subroutine library. The 'Hold' file is a separate part of memory. Neither BASIC nor RENUMBER can use this memory for their operation. Therefore, for maximum useable memory, use the hold file only while merging programs.

That's -30- for this month.

-Kathy Rudovsky

Those present at the February meeting of the Nuer's group took an in depth look at the DOS 3.3 Systems Masters for both the Apple][+ and the Apple][e. There isn't too much difference between the two, however, one noticeable difference is the lack of demo programs on the][e.

There is no implication here that demo programs belong on a Systems Master, just that it is a nice touch. The][e Master loads in considerably faster. For those impatient computer types (like me) it is a nice upgrade.

During the "Walk Through Your Computer" session, we found out some of the complications that occur when a disk drive is not operating properly, e.g. chews up diskettes and an incessant number of I/O errors. Reversing the drives on the card did help on the club's drives, but Disk Drive Hospital was the final diagnosis for Rhona and Armand.

The introduction of some Applesoft Basic statements hopefully stirred enough curiosity for those present to go home and try to write a little program. PRINT, INPUT, IF ... THEN, FOR ... NEXT, & STEP were examined. DOS Commands explored included PR#6, CATALOG, LOAD, LIST, and RUN.

Many thanks to Mike Iverson, author of Glossary One, for his time and patience in explaining how to send Control codes to the Gemini 10X printer to utilize the Elite Font through Apple Writer][. <CTRL> V ESC B <CTRL> b <CTRL> V. Mike says that he will be adding this feature and more to keep up with the ever expanding features of many different printers.

The major topic for April's meeting will be graphics and thus far a series of INTRO Seminars is still in the coordinating stage for May.

NUER'S rendezvous on the third Wednesday of each month at 7:30 pm at the SFAC Clubhouse, 1515 Sloat Blvd. #2. Anyone interested or just plain curious is welcome to attend. The atmosphere is informal and friendly.

Marketing Notes

-Don Scellato

If you are willing to assist the Club in the operation of its booth at the West Coast Computer Faire, look for sign up sheets at the February General Meeting. Anyone who signs up to help will be expected to work for about four hours at the booth. That time will be taken up in selling Club products and in signing up new members. Entry tickets for the show will be provided by the SFAC to everyone who helps at the booth.

The West Coast Computer Faire is scheduled to run from March 22 through March 25 at Brooks Hall at the Civic Center in San Francisco. Anyone who missed the sign up sheet at the February Meeting but who would like to help should call me at the number listed in the Cider Press. If any slots are left, I would be glad to fit you in. Even if all of the Club's slots are filled, we have been approached by other parties for assistance.

A briefing will be held at the Clubhouse on Saturday, March 10, at 10:30 A.M., to acquaint those people who have signed up to work at the booth with procedures, product, etc. For those members who don't want to help but who are looking for discount price tickets, we are attempting to find out from the new management of the Faire if tickets are available.

Look for more information in this column in the next issue of the Cider Press. Announcements will also be made at the February General Meeting. If any tickets are available, they will be sold on a First Come First Served basis at the March General Meeting.

If any member has written a program which he wishes to share with the rest of the members of the SFAC, bring it to the Sales Table or mail it in to the Clubhouse on disk along with any notes about how it works. If we include your program on a disk of the month, you get a free copy of the disk which includes your program or a copy of any other disk which is currently available at the Sales Table.

Starting in March we are going to try to issue at least one Disk of the Month (DOM) each month. These DOM's will contain a variety of programs which might be useful or entertaining to club members. We will also continue with the release of special applications disks such as the VisiCalc Template disks and Pascal disks as we receive material.

I would like to thank Joe Bianco, Arlene Strick, Chris Chung, Larry Nebron, and Steve Booker for their continuing assistance at the Sales Table each month. They contribute a lot of their time picking up product at the Clubhouse, setting up the table, selling product, breaking down, and taking everything back to the Clubhouse. If anyone else is interested in helping, see Joe Bianco at the Sales Table.

If anyone in the East Bay is interested in assisting with the production of Disks of the Month, other disks and disk paks, or working on documentation, please contact me at the next meeting or call me at the number listed in the Cider Press.

Two brands of disk drives are now available for club members at discount prices. The RGB Designs 35 Track full height drive, 40 track full height drive, and 35 track half high drive are all available. The 35 track drive is available for $220 plus tax. The other RGB designs disk drives are available for $235 plus tax.

Panasonic half high drives are now available for $190 to club members. Anyone who is interested in purchasing any of these disk drives should contact me at least one week prior to a meeting so I can order the drives for delivery prior to the meeting. The Panasonic drives will take slightly longer than the RGB Designs disk drives to arrive as the supplier is not in the Bay Area.

San Francisco Apple Core, Inc.

1515 Sloat Blvd., Suite 2, San Francisco, CA 94132

(415) 566-2342

The San Francisco Apple Core is a non-profit corporation whose purpose is to share information about Apple computers through member services, interaction and the club newsletter. The SFAC is run by volunteer officers appointed by an annually elected Board of Directors. Membership in the SFAC is not limited to Apple owners.

GENERAL MEETINGS are held monthly in Bldg. C, Fort Mason Center, San Francisco, CA (see map on reverse side) Meetings include demonstrations of products and techniques and are scheduled at 10:00 a.m. on the third Saturday of the month. (Occasional scheduling conflicts may occur - phone **clubhouse** for changes. (415) 566-2342) "Interest groups" meet at various locations in the S.F. Bay Area and focus on specific areas of Apple use i.e. Visicalc, Basic, Pascal, etc.

The **CIDER PRESS** is the official publication of the SFAC, published approximately 10-12 times a year, mailed bulk rate to members in the United States, and air mail to members in foreign countries (Canada - first class). The **CIDER PRESS** includes club news, program listings, tips, reviews and special features which are CONTRIBUTED BY MEMBERS.

PROGRAM DISKS are assembled from **PUBLIC DOMAIN, NON-COPYRIGHTED** programs donated by members. These disks may contain demonstration programs, utilities, games, educational programs and are available for purchase by members at the general meetings or by mail. Collections of these disks called LIBRARIES are also available from the club. Details regarding purchases of these and other items may be found in the CIDER PRESS.

TO BECOME A MEMBER, complete and return the application form below with the appropriate payment (U.S. Funds).

All applications must be in the name of a **person**. Company or organization names are not accepted.

The amounts listed below include the $20.00 annual dues plus foreign mailing costs where applicable.

Please allow 6-8 weeks for processing of your membership.

USA	$20.00	South America, Europe, North Africa	$41.00
Canada	$30.00	Eastern Europe, Asia, Pacific Isles	$47.00
Mexico, Carribean, Central America	$30.00	Middle East, Australia, South Africa	$47.00

Detach along dotted line

Application for Membership

SAN FRANCISCO APPLE CORE

FOR OFFICE USE ONLY		

Name _____

Address _____

City _____ State _____ Postal Code _____

USA ☐ Other _____

Phone Number (Home) _____ (Work) _____

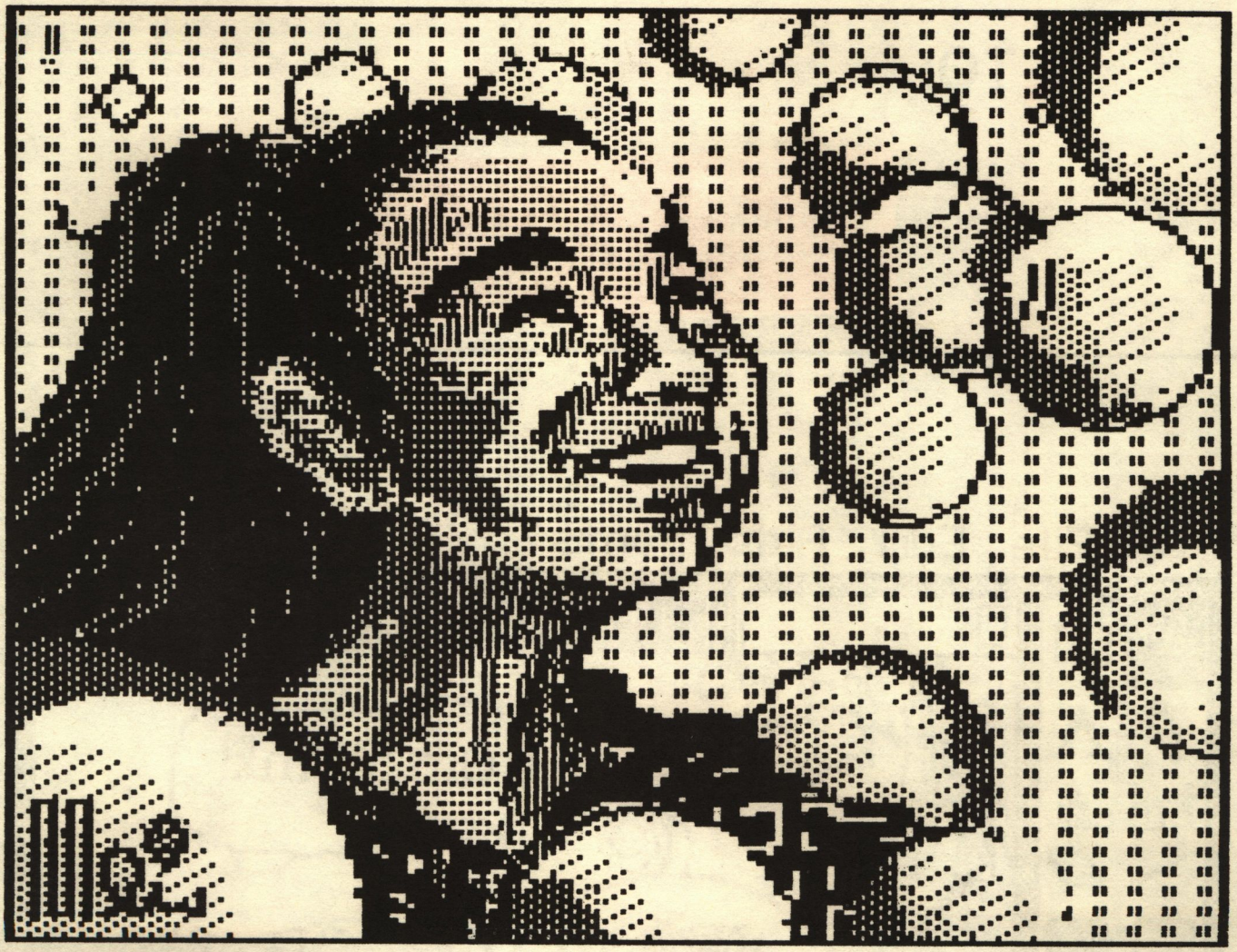
Check here if you have been a member previously ☐

Send application form and payment to:

S.F. APPLE CORE
1515 Sloat Blvd., Suite 2
San Francisco, CA 94132

Amount Enclosed $_____

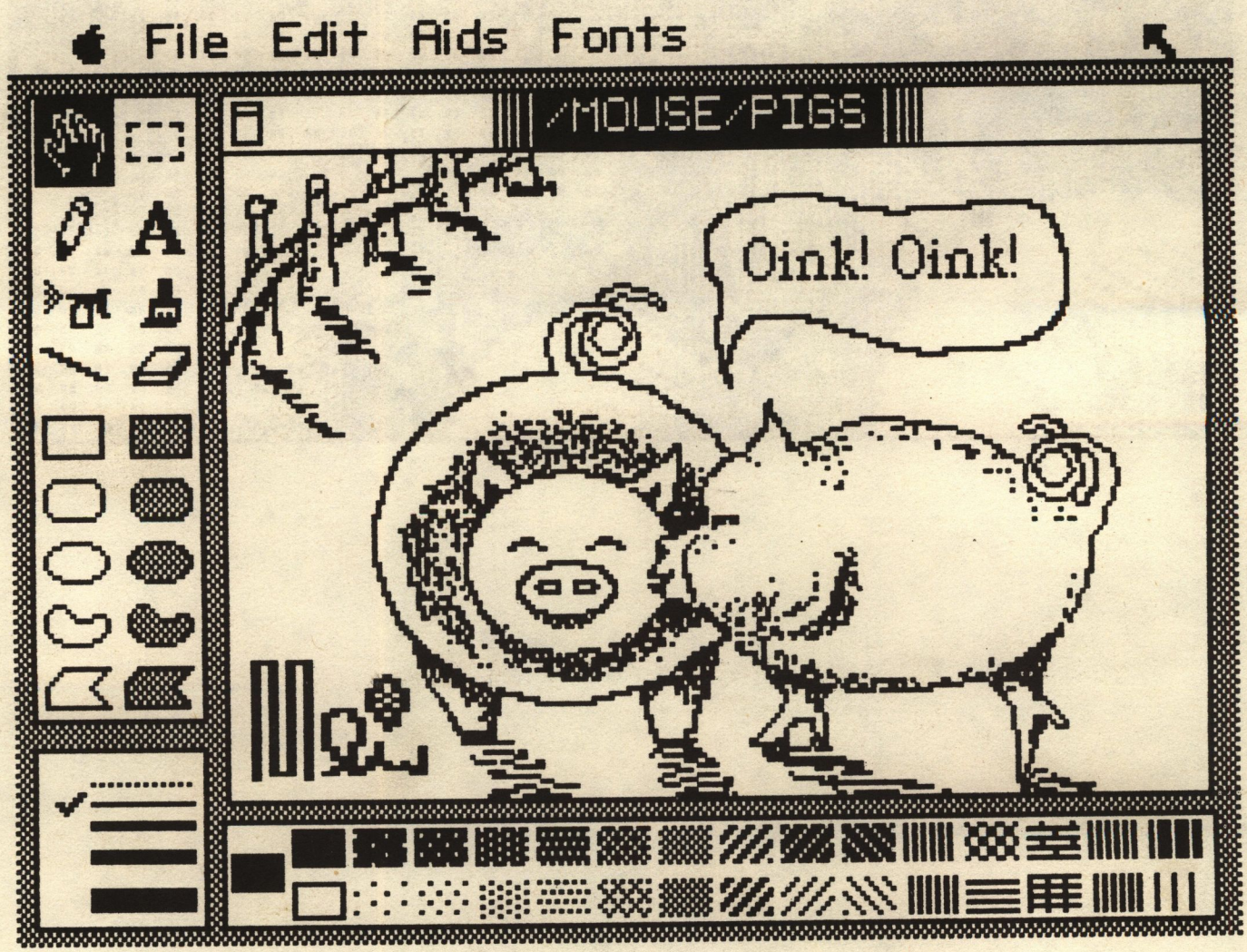
[Rev. 1/84]



SF APPLE CORE
1515 Sloat Blvd., Suite 2
San Francisco, CA. 94132

TIME VALUE

Bulk Rate
U.S. Postage
PAID
Permit #85
El Cerrito, CA 94530

TO:

3121AUG84
DOUGLAS STORKOVICH
921 PINTO CT
WALNUT CREEK CA 94596